



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

**The Motto of the University
(SEWA)**

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



**Bachelor Of Computer Applications (BCA)
Course Name: SOFTWARE ENGINEERING
Course Code: BCA-4-01T**

ADDRESS: C/28, THE LOWER MALL, PATIALA-147001

WEBSITE: www.psou.ac.in



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME & COURSE COORDINATOR :

Dr. Monika Pathak

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

PROGRAMME CO-COORDINATOR :

Dr. Gaurav Dhiman

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

BCA-4-01T: Software Engineering

Total Marks: 100

External Marks: 70

Internal Marks: 30

Credits: 6

Pass Percentage:40%

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short q questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Course: Software Engineering	
Course Code: BCA-4-01T	
Course Outcomes (COs)	
After the completion of this course, the students will be able to:	
CO1	Understand the software development life cycle which increases the growth opportunity.
CO2	Learn the detail knowledge of software requirement analysis.
CO3	Understands the detailed knowledge of software design and coding.
CO4	Understand the software testing that is relevant to the industry.
CO5	Acquire the detail knowledge of the fundamentals, including terminology; the nature and need for maintenance; maintenance costs and software evolution

SECTION-A

Unit I: Introduction of Software Engineering: The Problem Domain, Software Engineering, Challenges, Software Engineering Approach. Software development life cycle and its phases, Software development process models: Waterfall, Prototyping, Iterative.

Unit II: Software Process: Characteristics of software process, Project management process, Software configuration management process.

Unit III: Project Planning: Activities, COCOMO model. Software Metrics – Definition, Importance, Categories of metrics. Software Quality – Attributes, Cyclomatic complexity metric.

Unit IV: Software Requirements Analysis: Need for SRS, Data flow diagrams, Data Dictionary, entity relationship diagram, Characteristics and components of SRS, validation, metrics.

SECTION-B

Unit V: Software Design: Design principles, Module-level concepts, Structure Chart and Structured Design methodology, verification, metrics: network metrics, information flow metrics.

Unit VI: Coding: Programming Principles and Guidelines, Verification- code inspections, static analysis.

Unit VII: Software Testing: Testing fundamentals, Black Box Testing: Equivalence class partitioning, Boundary value analysis, cause-effect graphing; White Box Testing: Control flow and Data flow based testing, mutation testing; levels of testing, test plan, test case specification, test case execution and analysis.

Unit VIII: Software Maintenance: Categories of maintenance. Software Reliability – Definition, uses of reliability studies

Reference Books:

- Pankaj Jalote, “An Integrated approach to Software Engineering”, 3rd Edition 2005, Narosa Publications.
- K.K. Aggarwal, Yogesh Singh, “Software Engineering”, Revised 2nd Edition, New Age International Publishers.
- Roger. S. Pressman, “Software Engineering – A Practitioner’s Approach”, 5th Edition, Tata McGraw Hill.

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 1: INTRODUCTION OF SOFTWARE ENGINEERING

1.1. INTRODUCTION

1.2. PROGRAM

1.3. SOFTWARE

1.4. DIFFERENCE BETWEEN PROGRAM AND SOFTWARE

1.5. TYPES OF SOFTWARE

1.6. COMPONENTS/PARTS OF SOFTWARE

1.7. CHARACTERISTICS/FEATURES OF SOFTWARE

1.8. APPLICATIONS OF A SOFTWARE

1.9. ATTRIBUTES/PROPERTIES QUALITIES OF A GOOD SOFTWARE

1.10. SOFTWARE CRISIS

1.11. SOFTWARE MYTHS

1.12. SOFTWARE ENGINEERING

1.13. EVOLUTION OF SOFTWARE ENGINEERING

1.14. SOFTWARE ENGINEERING — A LAYERED TECHNOLOGY

1.15. ROLE AND RESPONSIBILITIES OF SOFTWARE ENGINEER

1.16. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

1.17. SOFTWARE DEVELOPMENT PROCESS MODELS

1.18. WATERFALL MODEL/LINEAR SEQUENTIAL LIFE CYCLE MODEL

1.19. PROTOTYPING MODEL

1.20. ITERATIVE ENHANCEMENT MODEL

1.21. SPIRAL MODEL

1.1 INTRODUCTION

Today, computers are everywhere. Almost all organizations are dependent on the computer. It helps in marketing, accounting, sales etc. Computer is a combination of hardware and software. Computer cannot work without software. Software helps hardware to perform various tasks. Software has changed the society and their work. There are many different software packages in the market. People choose software according to their need and facility. It helps people in many different ways, including saving time and money. For example, CRM (customer relationship management) software helps business to manage their customer relationships.

1.2 PROGRAM

1. A program is normally complete in itself and small in size.
2. It is used by the single user. Program lacks in proper documentation because single user is the author or single developer of the program. User makes little documentation according to his/her needs without using any format of documentation.
3. In a program, author is the only user of the program. If any problem occurs in the program and program crashes, then the author can fix the problem and can start using program again without any disturbance.
4. Program is the result of adhoc development where issues like portability and reliability are not considered.

1.3 SOFTWARE

Computers cannot perform any task/work without getting instructions from user through software. The combination of hardware and software is necessary if we want to get some task to be performed by the computer:

1. Software gives instructions to the computer what to do and how to do it.
2. Software is used for managing controlling and integrating the hardware components of the computer.
3. Software is not only the collection of computer programs. There is a difference between a program and software.
4. IEEE defines: ***“Software is a collection of computer programs, procedures, rules and associated documentation and data.”***
 - **Program:** It is a list of organized instructions when executed behave in a predetermined manner.
 - **Procedures:** It is a step by step instruction to achieve a desired result. It explains how to prepare something.
 - **Rules:** Rules are basically standards for actions/activities. Rule is basically a collection of guidelines to achieve something.
 - **Documentation:** It is a set of documents provided in a particular format. It is written on paper, online or digital media.
 - **Data:** Data is a raw material which is used in software. It is a set of values.
5. Software is basically developed by a team of developers who ensure the proper documentation.
6. Software developers ensure the well-designed interface because software are used by many people from different backgrounds.
7. Software is usually large in size and used by many people according to their needs.
8. Many resources and very many efforts are required for developing software.
9. Cost of developing software is very high because the software technology is still labor intensive and expensive.
10. Software projects are very large, involving many people and span over many years.
11. Software development is a systematic process where key issues like portability and reliability are considered.

1.4 DIFFERENCE BETWEEN PROGRAM AND SOFTWARE

Sr. No.	Concept	Program	Software
1.	Size	Size of program is small.	Size of software is large.
2.	Number of users	Single user	Large number of users.
3.	User interface	Program lacks in adequate user interface.	Software provide well designed and adequate user interface.
4.	Development	Program is the result of adhoc development.	Software is the result of systematic development.
5.	Number of developers	Single developer.	Team of developers.
6.	Documentation	User is the developer who makes little documentation according to his/her needs without using any format of documentation.	Software is the result of systematic work of team of developers. They provide proper documentation.
7.	Concentration on Issues	Issues like portability and reliability are not considered/concentrated.	Issues like portability and reliability are given proper concentration.
8.	Effort	Less effort is required.	More effort is required in the development of software as compared to program.
9.	Use of Resources	Less resources are required.	More resources are required as compared to program.
10.	Cost	Development cost of programme is less.	Development cost of software is approximately ten times more as compared to program.
11.	Complexity	Less complex.	More complex as compared to program.

Table 1.1 Difference Between Program and Software

1.5 TYPES OF SOFTWARE

Software is basically of three types:

1. System software
2. Application software
3. Service or utility software

1. **System Software:** System software is a set of one more programs designed to control the operation and extend the processing capabilities of a computer system.

The efficient system software allows application software to be run on the computer with less time and effort. System software makes the operation of a computer system more effective and efficient.

System software helps the hardware components to work together and provides support for the development and execution of application software.

Some commonly known system software are as follows:

(i) **Operating System:** An operating system is software that runs on computer and manages the computer hardware. Operating system performs basic tasks such as recognizing input from keyboard, sending output to the display screen, keeping tracks of files and controlling peripheral devices like printers and responsible for security ensuring that unauthorized users do not access the system. Operating system acts as an interface between hardware and software.

(ii) **Language Processors (Translators):** Language processors or translators are system software which transfer (convert) the instructions written in a programming language, into a form which can be interpreted and executed by a computer system.

It is of three types:

- **Compilers:** A compiler is a software that accept the total program code (high level language) as input and then converts it into machine code (low level language). Compilers also detect and indicate certain types of errors in source programs automatically.

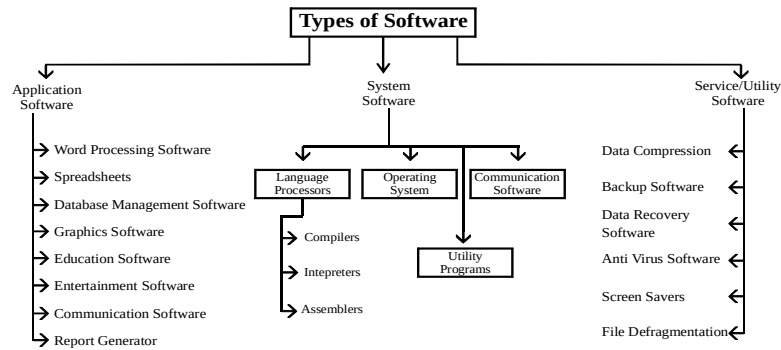


Fig. 1.1 Types of Software

- **Interpreters:** The interpreter takes one source program instruction, translates it into object code and executes it, then takes up the next instruction translates it and so on. Interpreters are slower than compilers.

- **Assemblers:** An Assemblers is a system software that takes as an input a program written in a assembly language and as an output generates the program written in machine level which can be directly executed on the computer.

(iii) **Communication Software:** Communication software enables to transfer the data and programs from one computer system to another computer system in a network environment where multiple computers are inter-connected together by communication networks.

(iv) **Utility Programs:** Utility programs is a set of programs which helps users in system maintenance and in performing tasks of routine nature like formatting of hard disks, taking backup of files, sorting records etc.

2. **Application Software:** Application software is a set of one or more programs, designed to solve a specific problem. There are wide ranges of applications ranging from simple applications like word processing, banking etc. to complex scientific/engineering applications like weather forecasting, design of aircrafts etc.

Some commonly known application software are as follows:

(i) **Word processing software:** Word processing software helps in day to day documentation work. It helps in creating texts, manipulating, formatting and printing of the text. It is considered as a computerized version of typewriter. It allows users to make changes and corrections in the text easily.

(ii) **Spreadsheets (Electronic Spreads Sheets):** Electronic spreadsheets are sheets of paper with rows and columns. Data is entered in a tabular form. The area where row and column meet is known as cell. User put text, number or formulae into cell. It also provides flexibility to user to make changes in spreadsheets. It provides various built in functions for the convenience of user.

(iii) **Database Management Software:** Database management system is a software package that allows a user to perform various operations like creating, deleting, updating, adding, modifying data in databases. Whereas database is an organized collection of logically related data. It makes retrieval of information easy and effective.

(iv) **Graphics Software:** Graphics software is a collection of programs which manipulate the images to any extent. These programs are referred to as point or draw programs in which user can draw illustrations from the scratch using an electronic pointing devices such as mouse, light pen or brush. Graphics programs are also known as point programs.

(v) **Education Software:** Education softwares are known as teaching and learning tool. These softwares are used to teach students. These softwares are very effective in mathematics, recognizing of alphabets, read whole words and sentences etc.

(vi) **Entertainment Software:** Entertainment software allows a computer system to be used as an entertainment tool. A good example of such an application is computer video games.

(vii) **Communication Software:** Communication software is used to share information and resources by connecting the computers. Communication software is also known as network software. A user can share information with another user with communication software.

(viii) **Report Generators:** Report generator helps a user to design his own report. This method is very easy and less time consumed. Manually design of these reports is very difficult task. Commonly available report generating software is oracle reports.

3. **Service or Utility Software:** Utility software is used to provide services to facilities the working of the operating system. It helps user to take backup of data, scanning virus and retrieval of deleted files.

Some commonly known utility software are as follows:

(i) **Data Compression:** Data compression is a ability to reduce the storage requirements of a file using mathematical algorithms. It helps to store more data on a disk. Commonly used data compression software are easyzip and WinZip.

(ii) **Backup Software:** The backup software helps the user to copy large group of files from hard disk to other storage media like floppy etc.

(iii) **Data Recovery Software:** It is also known as unerase software. It is required when user delete a file accidentally from a disk. After deleting a file, user realize that file is still needed and very important. This software gets back those deleted files which were deleted by the user.

(iv) **Antivirus Utilities:** The antivirus software track the virus, eradicate and prevent their spread. The antivirus scans the disk, identify any virus and attempts to remove them. Examples are McAfee, Norton antivirus etc.

(v) **Screen Savers:** Screen saver displays moving images on the screen if no input is received from the user for several minutes. A variety of screen savers are available ranging from flying windows to hollywood personalities and desktop themes.

(vi) **File defragmentation:** A file defragmentation is used to defragment the files to speed up the working of the hard disk. It is now part of the window operating system.

Comparison between system software and application software

Sr.No	System Software	Application Software
1	It is general purpose software.	It is specific purpose software.
2	System software is a superset of application software.	Application software is a subset of system software.
3	Examples: operating system, language processor, communication software.	Examples: word processing, software database management system, spreadsheets etc.
4	System software gets installed when the operating system is installed on the computer.	Application software gets installed according to the requirement of the user.
5	System software can run independently without the presence of the application software.	Application software cannot run without the presence of the system software.
6	It is essential for a computer.	It is not essential for a computer.
7	The number of system software is less than application software.	The number of application software is much more than system software.

Table 1.2 Comparison between system software and application software

1.6 COMPONENTS/PARTS OF SOFTWARE

The components of a software means those parts which complement a software product. Software needs other existing softwares for communication. These parts are necessary for the software development. The combination of these parts make a complete software.

The following are components of a software:

1. **Operating Environment:** Operating environment describe the suitable environment where software will operate on particular hardware platform in a peaceful manner.

2. **User classes and characteristics:** Various user classes are used to anticipate the software product. User classes are based on many aspects like frequency of use, technical expertise, security levels, educational level etc. The characteristics of user classes are also described in this section.

3. **Documentation:** Documentation is the text in the written form, which explains all the operating mechanisms/methods, functions and uses of the software. It provides the full-fledged information about the software. It explains complete information of the software in a written form. It is of three types:

(i) **Architecture/Design Documentation:** This document gives an overview of the software. This type of document gives the justification of the list things which are used in the development of the software. The design documentation also explains why such things are used in the software.

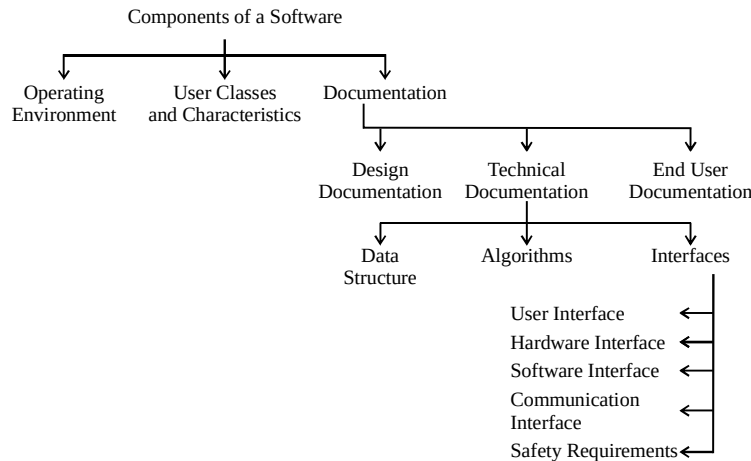


Fig. 1.2 Components of a Software

(ii) **Technical Documentation:** The technical documentation provides the coding part and various aspects of operations of the software. This documentation also explains the data structures, algorithms and interfaces:

- **Data Structure:** A data structure is a systematic way of organizing and accessing data. It explains how data are stored in a computer.
- **Algorithms:** An algorithm is a procedure to solve a problem. The algorithm is the basic technique used to get the job done.
- **Interfaces:** It is a boundary across in which two independent system meet and act on to communicate with each other. The following are the types of interface:
 - (a) **User Interface:** It explains the characteristics of interface between the software product and users. It include the screen images, GUI standards, standard buttons and functions, keyboard shortcuts, error messages etc.
 - (b) **Hardware Interface:** It explains the characteristics of the software product and hardware components of the system. It include the supported device types, nature of data, communication protocols, control interactions between the software and hardware.
 - (c) **Software Interface:** It explains the connection between the software product with other specific software components, databases, tools, operating systems, messages coming into the system and going out. It also explains the communications nature and services needed.
 - (d) **Communication Interface:** It explains the communication functions of any product like email, communication protocols, electronic forms etc. It also describes the used communication standards like FTP or HTTP.
 - (e) **Safety requirements:** Safety requirements are concerned with damage, harm or loss which occurs during the usage of the product. It also explains some actions that prevents from these damages.

(iii) **End-User documentation:** This documentation helps the end users, system administrators and support staff in understanding the software. It provides all the features and steps required in the software. In short, it is a type of guide tutorials or a reference guide for a beginner. It should be very simple, manual and understandable so that end-user can easily understand it and use it.

1.7 CHARACTERISTICS/FEATURES OF SOFTWARE

The uses of computers are growing very rapidly. The computer industries are also growing very rapidly. The computer industries are very different from manufacturing industries in many aspects. Manufacturing industries are those industries who make/manufacture T.V.s, buildings, bikes etc. Computer industries develop the software not manufactured. Software is a logical entity not a physical entity. User cannot touch the software, cannot feel the quality of software.

The following are the characteristics of a software:

1. **Software is not manufactured or created, it is developed:** Software is first designed and then developed. Software is a logical entity, so it is developed not manufactured. Industries manufactured only those products which are in physical form, which can be touched. Software is not in a touchable form that is why it is developed.

Developers have number of automated case tools but quality of software depends upon the skills and creative abilities of the individual developers. Developers need managing and controlling abilities to ensure the quality of the software product.

2. **Software is Highly Malleable:** Software is different and highly malleable as compared to other products. For example, we buy a new cabinet for old T.V. but old T.V. does not fit into this new cabinet. What we do? We call the carpenter to make necessary changes in the cabinet so that T.V. can fit into the new cabinet. On the other hand, if we want to change something in the software. We can directly modify the software itself rather than changing in the design of the software.

3. **Software does not wear out but deteriorates:** Software does not wear out means software is not influenced by the environmental changes. Software is a logical entity. Software functions exactly the same way years after years without changing requirements over a period of time. If changes are necessary in the software to meet the user's new requirements, then there is a possibility of occurring of some defects which deteriorates the quality of the software.

Manufacturing companies replace the defected parts with the spare parts and maintain the quality of the product. But this cannot happen in the software industries. Software cannot replace with the spare parts.

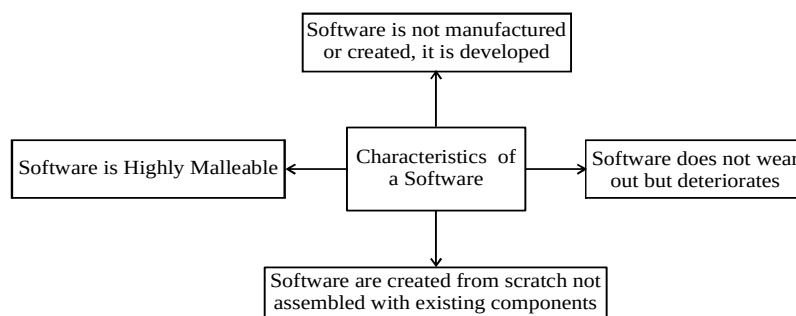


Fig. 1.3 Characteristics of a Software

4. **Software are created from scratch not assembled with existing components:** In manufacturing companies, various identifying components are put together to form the original masterpiece. Every component is flexible and independent in nature. But this approach is not applicable in software development. In software development approach, each module is inter-linked with another module. Each module is further divided into sub-modules. Different development teams are required for

different modules and sub-modules. Here modules are not assembled, modules are developed and inter-linked with each other to make a complete software.

1.8 APPLICATIONS OF A SOFTWARE

Software are used in many areas such as business, entertainment, medical, reservations etc. There is no area where software is not used.

The following are some areas where software are used:

1. **Business Softwares:** Softwares are used in business for fast, efficient and intelligent functions/tasks. With the help of softwares, companies make charts, excel sheets etc. which help in effective decision making. Companies also use software for visuals and advanced formatting.

2. **Entertainment Softwares:** Softwares are also used for entertainment purposes. There are number of entertainment softwares in the market. These softwares are extremely popular over the past decades. These softwares includes games, graphics, multimedia etc.

3. **Scientific Investigation and Engineering Software:** These softwares satisfy the requirements of scientists and engineers. These software are specially developed and designed for scientific and engineering purposes. These softwares are capable to perform complicated algorithms, programs etc. Astronomy and satellite launching etc. are related to scientific investigation and engineering softwares.

4. **Education Softwares:** Education softwares are known as teaching and learning tool. These software are used to teach students. These softwares are very effective in mathematics, recognizing of alphabets, read whole words and sentences etc. The primary purpose of education software is teaching or self-learning.

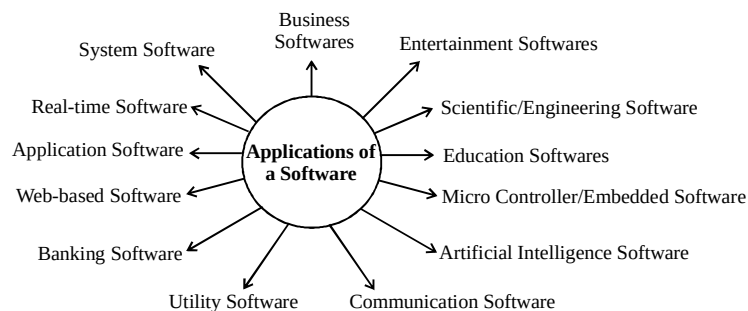


Fig. 1.4 Applications of a Software

5. **Micro Controller/Embedded Software:** Embedded software is designed for a particular kind of application service. It performs one or more predefined tasks. It is a combination of computer hardware and software. Embedded software refers to computer programs that directly interact with hardware. Example — Cars, Microwave, Watches etc.

6. **Artificial Intelligence Software:** Artificial intelligence software is an advance technique software. These softwares can understand a natural language, find out the reasons and their solutions automatically around the world. Artificial intelligence software requires human type of intelligence. Example—Expert system, Pattern recognition etc.

7. **Communication Software:** Communication software is used to share information and resources by connecting the computers. A user can share information with another user with communication software. With the help of modem and telephone line, a person can make contact with another person located at any place in the world.

8. **Utility Software:** Utility software is designed to help in management and tuning of computer hardware, operating system and application software. It is normally used to solve the common problems of software and hardware. The examples of Utility software are as follows:

- Disk defragmenters
- System profilers
- Virus scanners

- Application launchers
- Network managers
- Encryption utilities.

9. **Banking Software:** Banking software is an enterprise application solution that enables Banks to provide banking services and content to customers with efficiency and reliability within their bank premises. The banking software provides following features:

- Customer Management
- Transaction Management
- General Ledger & Accounting
- Accounting Transaction /Maintenance
- Rule Implementation / Maintenance
- Control / Configuration Management
- User-orientation
- Reporting
- Security
- Control / Configuration Management

10. **Web-based Software:** Web based software is an application package that can be accessed through the web browser. The software and database reside on a central server. Web based application gives us an opportunity to access the business information from anywhere in the world at anytime. It also facilitates us to save time, money and improve the interactivity with the customers and partners. It is also widely used for information exchange, entertainment and e-commerce.

11. **Application Software:** Application software is a computer software that makes use of the capabilities of a computer directly and thoroughly to a task that the user wishes to perform. Typical examples of software applications are word processors, spreadsheets and media players. Multiple applications bundled together as a package are sometimes referred to as an application suite.

12. **Real-time Software:** The term “real-time” refers to the ability to do or perform tasks right at the very moment they are said to be executed. Real-time software enables the user to execute various tasks and activities at the same time, as long as the programs are kept open. In computer systems, real-time operating systems contain a multitude of programs to run and operate even if the user is focused only on just one application. These software include media, building tools, computing and analysis applications, and system maintenance

13. **System Software:** System software is a type of computer program that is designed to run computer hardware and application programs. The operating system (OS) is the best-known example of system software. The OS manages all the other programs in a computer. Other examples of system software are BIOS (basic input/output system) program, the boot program, assembler, device driver, etc.

1.9 ATTRIBUTES/PROPERTIES QUALITIES OF A GOOD SOFTWARE

Quality of a software depends upon different parameters. No single property fully describes the quality of a product. A software is called a good quality software if it possesses the following attributes:

1. **High Quality:** A good software product should possess the high quality means software should have minimum number of errors. The minimum number of errors make software more reliable.
2. **User Friendly:** Software should be user friendly. It should be usable by all types of user. Software should be easy in use. Software should provide all appropriate and adequate user interface along with the sufficient documentation.
3. **Secured and Safe:** Software should not cause any economic/physical damage. It should be secured and safe to work without any failure.
4. **Fit for intended job:** Software should fulfill all the requirements of the user. It should work according to the needs of the user and should be fit for the intended job. A quality software always fulfils the purpose of the user.
5. **Within the budget:** Software should be within the budget for the user. The cost of software should be within the packet of the user. Software should give profit to the computer industry.

6. **High Adherence (tolerance) power:** Software should respond to each and every type of input of the user. Software should behave reasonably even in the anticipated situations.
7. **Maintainable/Modifiable:** A good software should be maintainable and modifiable according to the situation. It should be changed to meet the new requirements of a user/customer.
8. **Efficiency:** Software should utilize resources in such a way that there should be no wastage of any resource. For example: memory and processor should be utilized up to the optimum level.

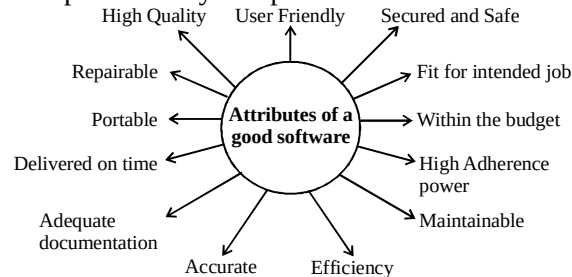


Fig. 1.5 Attributes of a good software

9. **Adequate documentation:** A software should have adequate documentation so that it can solve its purpose very well.
10. **Accurate:** Software should perform all tasks very accurately. Software should perform all operations according to the predefined specifications of the user and give accurate output to the user.
11. **Delivered on time:** Software should be developed and delivered within the specified time.
12. **Portable:** Software should be portable and have ability to work in different platforms.
13. **Repairable:** Software should be easily repairable. It should have privilege to correct the errors in a limited time period with the small amount of effort.

1.10 SOFTWARE CRISIS

Development of large software product with many essential features is the manual, error prone and complex process. The term software crisis is used in development of software. The term software crisis refers to the difficulty in developing a software like writing correct, understandable and verifiable computer programs. The main reasons of term software crisis are expectations, level of complexity and changeability. For example: every user demands large number of features in a required time and in a minimum budget. These expectations take the software into the software crisis.

“Software crisis means difficulty in writing correct, understandable and verifiable computer program.”

Software crisis are the problems encountered in the development of the software. This is due to many issues like projects are not managed properly, late delivery of software, exceeds the budget for the development of software etc.

The following are the main reasons of software crisis:

1. Lack of communication between developers and users.
2. Lack of understanding of the problem.
3. Lack of adequate skills in the development staff.
4. Large and complex problems.
5. Increase in cost of software as compared to hardware.
6. Lack of adequate environment for developing the software.

The following are the some software crisis which are faced by many software industries:

1. Software is expensive.
2. Late delivery.
3. Software is not as per user requirements.
4. Difficult to modify/change and rework.

5. More chances of failures.
6. Brittleness.
7. Inadequate documentation.
8. Low quality of software.

In short, many software projects failed in late 1960's because these projects were late, over budget, complex, difficult to maintain and did not satisfy the client's requirements. This was known as software crisis.

Companies has been made many improvements and changes to solve these problems but problems are not decreasing. After using various methods and processes, companies are still facing new challenges and problems with the same old problems. Companies come with the conclusion that no single approach prevents the project/software over runs and failures in all cases.

In general, large complicated and poorly documented software projects have large, unanticipated problems. Whereas small projects have small problems. Researchers are still searching for the solution of the crisis.

1.11 SOFTWARE MYTHS

The term myth relates with mis-information or confusion. There are so many myths associated with software development. These myths affect the development of software indirectly. It means because of these myths, problems are arise in the initial stage of software development. There is no truth behind these myths. Some myths related to software development are as follows:

1. Software is good if it has many features.
2. Many software engineers cover up the delay.
3. Quality of software can be checked only after execution.
4. User requirements can be changed during any stage of software development process.
5. Reusing software increases safety.
6. Testing software can remove all the errors/bugs.
7. Software is easy to change.

1.12 SOFTWARE ENGINEERING

1. Software engineering has different meaning and definition for many people. People explain software engineering according to their interest whereas experts give different definitions.

According to IEEE

“Software engineering is a process of systematic and disciplined approach for the development, operations and maintenance of the software.”

According to Barry Bohem:

“Software engineering is the application of science and mathematics by which the capabilities of computer equipment we made useful to man via computer programs, procedures and associated documentation.”

2. Software engineering ensures the development of good quality software in a minimum budget. It is the disciplined process which is concerned with all aspects of software production.

3. Software engineering follows quality standards like ISO 9001 and capability maturity model to maintain the quality in software. It focuses on various methods which are used in many software projects.

4. **Need of Software Engineering:** The goal of software engineering is to develop methods and procedures for software development in a consistent manner to produce high quality software at low budget. Software engineering follows quality standards like ISO 9001 and capability maturity model to maintain quality in software. To achieve this goal, software development should be done in phases. Software processes should be designed and controlled properly to maintain consistency in development of software. The aim of the software engineering is to take the development nearer to the science and to concentrate on methods helpful in software development.

5. **Issues in software engineering:** The following are some issues/problems related to software engineering.

- **Technology and Management:** Developing a large software is very difficult as compared to small software. Large software needs adequate technology (methods, procedures and tools) and proper management so that software can be developed in a systematic manner in a required time without facing any problem.

Large projects are tightly managed to control the cost schedule and quality. Formal technology requirements of large software projects are high.

Whereas, small software projects can be developed with informal methods and technologies. For example: if we count the people in a room. It requires easy method and less time. But if we want to take a census of a country, then it requires proper formal methods and procedures for management and organization.

- **Consistency in terms of performance and quality:** Consistency of performance and quality is very important for all computer industries. The aim of computer industries is to develop a software with low budget and high quality in a consistent manner. Computer industries want consistent quality with improved productivity but it is very difficult task. A good quality software in a low cost is a challenge for computer industries.

- **Expensive:** Software development needs resources like hardware, software, manpower etc. in a low budget. But it is a challenge because software development is a labor intensive, time consuming and costly method.

1.13 EVOLUTION OF SOFTWARE ENGINEERING

Evolution of software engineering means take a look to check how software engineering been over the years. With the passage of years, scope and meaning of software engineering enhanced with number of issues.

During 1960, software was developed by a single programmer or by a small team of programmers. In this period, programmer can draw flow diagrams using language tools, write programming codes according to his/her choice. In 1970, team of programmers became larger on the basis of type of software project. Cost estimation was not considered but programming languages like FORTRAN and quality assurance got introduced. During the period of 1980's and 1990's, standards and methodologies were introduced for ensuring quality and reducing cost in software product.

Computer aided software engineering (CASE) companies use no of tools (Knowledge Based Systems and Total Quality Management) to improve the software development process.

1.14 SOFTWARE ENGINEERING — A LAYERED TECHNOLOGY

Software engineering follows the layered technology which are as follows:



Fig. 1.6 Software Engineering — A Layered Technology

1. **Methods:** It provides technical support for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support.

2. **Tools:** Tools provide automated or semi-automated support for the process and methods.

3. **Process:** It is a framework with activities for effective delivery of software engineering technology. A process is a collection of activities, actions and tasks that are performed when some work product is to be created. The purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

4. **Quality Focus:** Any engineering approach must rest on organizational commitment to quality which fosters a continuous process improvement culture.

1.15 **ROLE AND RESPONSIBILITIES OF SOFTWARE ENGINEER**

An engineer is someone who uses advanced knowledge of science, mathematics, and technology to build objects for use by others. Whereas software engineer is responsible for the complete life cycle of a new/modified software product from design to implementation. The roles and responsibilities of software engineer are as follows:

1. Software engineer is a good programmer and have a good knowledge of data structures and algorithms. Software engineer is expert in more than one programming language and have creative or logical thinking mind.
2. Software engineer is familiar with many design approaches and have ability to move among several levels of abstraction of different stages of the project, to a specific design for system and at the end to the detailed coding level.
3. Software engineer consult with systems analysts, programmers and others to collect the limitations, capabilities and performance requirements of software product.
4. Software engineer modify the existing software to correct errors, allow it to adapt to new hardware, or to improve its performance.
5. Software engineer analyze the user needs and software requirements to determine feasibility of design within time and cost constraints.
6. Software engineer coordinate software system installation and monitor functions to ensure specifications are met.
7. Software engineer analyze information to determine, recommend, and plan computer specifications and layouts, and peripheral equipment modifications.
8. Software engineer supervise the work of programmers, technologists and technicians.
9. Software engineer utilizes software engineering tools such as configuration management systems, build processes, and debuggers in the software development process.
10. Software engineer serve as a mentor to less experienced software engineers.
11. Software engineer also provides training to new users and handles support and feedback.
12. Software engineer produce well-organized, optimized, and documented source code. Contribute to technical design documentation.
13. Software engineer continuously learn and improve skills and can work independently when required.

1.16 **SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)**

Software development life cycle is also known as product development life cycle (PDLC). It is also called software development process (SDP). Software development life cycle has many phases and activities for software development. Each phase ends with a defined output which is the input for the next phase.

The goal of SDLC is to develop a high quality product in minimum cost by checking the progress at the end of each phase. In all the phases, complexity of tasks and project tracking etc. should be managed to maintain the quality of the software product. It should also help to solve the software crisis.

SDLC varies with large projects and small projects. Small projects have different phases, activities and issues as compared to large software projects. Large software projects should require proper documentation with user specifications whereas small softwares can be developed without written record of activities.

Key Points:

- In the SDLC, basically the major activities related to software development are performed.

- SDLC is a sequence of steps/phases to perform the well-defined activities to fulfill the desired goals. In SDLC output of one step/phase is the input of next step/phase.
- There is a criteria for each input/output phase. Input criteria specifies the conditions when to terminate the activities of the phase output criteria specifies the conditions when to terminate the activities of the phase.
- The goal of the SDLC is to detect the errors/defects in the starting of their origin in the phase. Validation and verification should be performed at the end of the phase to detect the errors.
- Due to importance of development process, various models have been proposed for developing a software which are being discussed in the this chapter. [2.3]
- Clear objective is defined in the starting of the development process to make it more effective and less expensive.

There are seven phase of SDLC:

1. Feasibility Analysis
2. Requirement Analysis
3. Design
4. Coding
5. Testing
6. Implementation
7. Maintenance

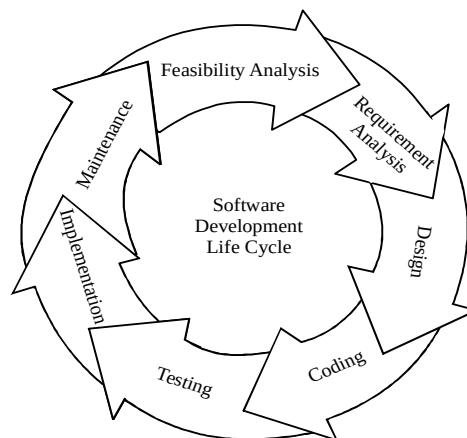


Fig. 1.7 Phases of SDLC

1. Feasibility Analysis/Feasibility Study:

- **Requirements Gathering:** The Business Analyst (BA) or a representator of a company collects all the predefined information of the company like what type of softwares a company can develop and in how many days etc. and goes to the client. Then BA collects all the basic requirements and specifications of the client and prepare a document known as BDD (business development design), BRS (business requirement specification), URS (user requirement specification), and CRS (customer requirement specification) etc. All these documents are same, only name is different. These documents are the output of the feasibility study phase and input of the requirement analysis phase.

• Types of Feasibility Study:

(a) **Technical feasibility study:** In the technical feasibility study, possibility of project development is measured. It is checked that whether the software project is technically sound or not.

(b) **Economic feasibility study:** In the economic feasibility study financial matters are discussed. For example : Is the software project is under budget or not? A software project should be under budget.

(c) **Schedule feasibility study:** In the schedule feasibility study, time duration is decided. The company finds the probability to complete the project in fixed allowed time period.

(d) **Motivational feasibility study:** In the motivational feasibility study, the sufficient motivation of organization is checked with necessary user participations, resources etc.

2. **Requirement Analysis:** The output of the feasibility study phase CRS (customer requirement specification) is used as input in the requirement analysis phase. Generally project manager analyze the requirements of the user and prepare the project plan.

Requirement analysis phase has 4 steps which are as follows:

(a) **Requirement Analysis:** All the requirements of client (mentioned in CRS) are studied and analyzed in detail. To understand the requirements of clients for large projects with many features, different tasks are analyzed. But it is very difficult because of communication gap between client and developer. This type of difficulty is based on basically two aspects. In one aspect, developer tries to develop a software which meets client's requirements without understanding the client's problem domain. In second aspect, client does not aware about the issues/problems in the development of the software.

(b) **Deciding Technology:** In this step, suitable technology is being declared for the development of software project. Companies select suitable technology based on many factors like requirements of clients, type of software project etc.

(c) **Estimating the Resources:** In this step, resources like manpower, time, money etc. are estimated for the development software project. All resources should be examined and improved in advance to reduce the difficulties in the software development.

(d) **Preparing SRS (Software requirement specification):**

- SRS (Software Requirement Specification) is a last and fourth step of requirement analysis phase. It is a document with the specific language to specify all the inputs, outputs, functional requirements, performance requirements with some constraints like economic issues, security etc.
- Definition: SRS is the specification for particular software product that perform certain functions in a specific environment.
- SRS is a complete description of the software which is going to be developed. It describes the complete scope of the product.
- SRS is prepared by Senior Analyst (SR).
- It is a type of agreement between the client and the company on what the software product is going to do.
- All functions/tasks of software development are based on the software requirement specification.

3. **Designing:** In the design phase, a plan is proposed for software development according to the software requirement specification. The quality of a software depends upon the design document. Design document is like a blue-print or plan for the software development which is used in later stages.

The design activity of a software development is divided into two levels.

(a) **High Level Designing/System Design**

- It is also known as top-level design. It is done by Technical Manager (TM).
- In this designing, software is divided into number of modules. Modules are identified along with their specification and interaction among them. In this type of designing, focus is on identifying the modules.

(b) **Low level designing/detailed design:**

- It is also known as detailed design. It is done by Team Lead (TL).
- In this designing, modules are further divided into number of sub-modules.
- Internal logic of sub-modules are specifies in the high level descriptive language.
- The output of this phase is detail design document (DDD) which is further used in later phases to develop a high quality software.

4. **Coding:**

- Once the designing of software is complete, the coding phase comes, in which developers follow some coding standards for writing the programs.
 - In this phase, design of a software product translates into simple and understandable code in a particular language.
 - Developers prepare the source code which is easy to write, read and understand.
 - Coding should be done with simplicity, clarity and well-written format. Effective coding can reduce the testing and maintenance effort.
 - Coding part is very time consuming and difficult task. Coding should be done in structured programming which linearizes the control flow in the programs.
5. **Testing:**
- The quality of a software product is measured and controlled by the testing phase.
 - Testing is a process of executing a program with the intent of detecting or finding errors. Proper planning is required for the testing.
 - Testing is very difficult and time-consuming process.

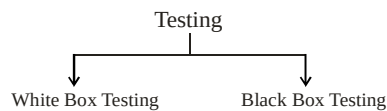


Fig. 1.8 Types of Testing

- Before starting this process, test engineers study the CRS (customer requirement specification) document prepared in the first phase and prepare a review report.
 - In the review report, test engineers mentioned those points which are not clear and understandable. Test engineer send this review report to the Bussing Analyst (BA).
 - Test Engineer prepare test plan before testing. Test plan specifies the scope, approach, resources and schedule of all testing activities.
 - Test plan covers all the activities like items to be tested, types of testing to be performed, features to be tested, persons/resources responsible for testing, time scheduling issues/risks associated with the plan etc.
 - Testing starts with the unit testing, in which a module is tested separately then integration testing is performed, which detect design errors and in last, system testing is performed to check the system performance according to the software requirement specification (SRS).
 - The output of the testing phase is known as defect profile document (DPD). Test Engineers sends this document to the team of developers for removing the errors and controlling the quality of software.
6. **Implementation:**
- In this phase, after completing the software project, a mail is sent to the client. This mail is known as software delivery note.
 - It after receiving this mail, client test the software project known as user acceptance testing. After this successful testing, software is installed in the client's environment.
7. **Maintenance:** During installation of software product in the client's environment, if any problem occurs or if client wants to make some changes in the software, then the maintenance people make changes to solve the problem and prepare the deployment document (DD) for further use.

1.17 SOFTWARE DEVELOPMENT PROCESS MODELS

1. A software development life cycle models describe the necessary sequence of different activities helpful in software development.
2. Software development starts with the request of client and undergoes many stages until software product is developed and delivered to the client.

3. These models show series of identifiable activities of software development for the successful delivery of the software product.
4. In the SDLC models, several activities and several documents are carried out. Documents are normally made to collect all the input/output information. Different phases of software development are shown graphically with the textual description for easy understanding.
5. In short, these models give descriptive and diagrammatic representation of software development phases.
6. There are so many software development life cycle models like waterfall model, spiral model etc. The common aim of these models is to develop a good quality software.
7. No matter which life cycle model is followed, all models have basic activities, carrying process of these models may be in different orders, but goal will remain same.
8. These models are helpful in developing a high quality software with low budget and time constraints.

There are so many life cycle models. The following are some life cycle models commonly used by computer organizations:

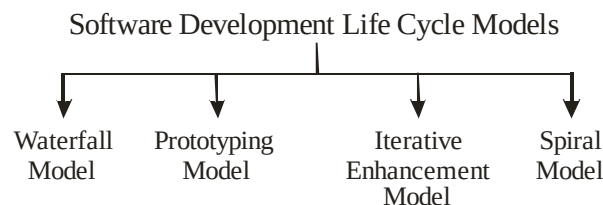


Fig. 1.9 Models of SDLC

1.18 WATERFALL MODEL/LINEAR SEQUENTIAL LIFE CYCLE MODEL

1. Waterfall model is the simplest and oldest process model.
2. It is represented by sequence of different stages.
3. The output of one stage flows into the second stage, the output of second stage flows into third stage and so on.
4. It is a definite structure of software development life cycle. Each phase is distinct and has a well-defined entry.
5. There is a provision for verification, validation and error correction at the end of every phase.
6. The part of waterfall model between the feasibility study and system testing is known as development part. After the development part, the software product is delivered to the customer. The maintenance phase comes after the delivery of the developed product to the customer.
7. This model is also known as the linear sequential model or the software life cycle model.
8. The waterfall model consists of following six stages:

(i) **Feasibility Study:** The feasibility study determines whether the developing model is financially and technically feasible or not. It is necessary to analyze the problem. More we understand the problem and the better we can identify alternative solutions. The feasibility study is usually done within limited time bounds. The outcome of feasibility study is a document that should contain at least the following factors:

- A definition of the problem.
- Determination of technical and economic viability.
- Alternative solutions and their expected benefits.
- Requirement of resources and detail about delivery

This report is called as feasibility study which is prepared by a group of software engineers. The client or the customer is also consulted through questionnaires. The report determines whether the project is feasible or not.

(ii) **Requirement Analysis and Specification:** In this phase, we exactly analyze the requirements and needs of the project. The purpose of a requirement analysis is to identify the qualities required for the application, in terms of functionality, performance, ease of use portability, and so on. The result of this phase is known as the software requirement specification (SRS) document.

An SRS document must contain the following items:

- Detailed statement of problem.
- Possible alternative solution to problem.
- Functional requirements of the software system.
- Constraints on the software system.

The SRS document must be precise, consistent, and complete and covers features like problem statement, introduction to the problem, functional requirements of the system, non-functional requirements of the system, behavioral descriptions and validation criteria.

(iii) **Designing:** The objective of the design phase is to convert the requirements specified in the SRS document into a structure that is suitable for execution in some programming language. Two different design approaches are available: the traditional design approach and the object-oriented design approach.

- In Traditional Design Approach, two activities are carried out, first a structured analysis of the requirements specification and second structured design. During structured design, the results of structured analysis are transformed into the software design.
- In Object-Oriented Design Approach, the various objects and their relationships within the system are identified.

(iv) **Coding and Unit Testing:** Coding and unit testing is the phase in which we actually write programs using a programming language. The output of this phase is an implemented and tested collection of units. The unit testing is the testing of code to check correctness and remove the bugs. The debugging is a related activity performed in this phase.

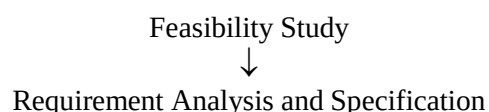
(v) **Integration and System Testing:** During the integration and system testing phase, the modules are integrated in a planned manner. The different modules are integrated to develop a software product. During each integration step, the partially integrated system is tested finally, when all the modules have been successfully integrated and tested, system testing is carried out. The objective of system testing is to determine whether the software system performs per the requirements mentioned in the SRS document.

The system testing is done in three phases: Alpha, Beta, and Acceptance Testing.

- **Alpha Testing** is conducted by the software-development team at the developer's site.
- **Beta Testing** is conducted by a group of friendly customers in the presence of the software-development team.
- **Acceptance Testing** is performed by the customers themselves. If the software is successful in acceptance testing, the software product is installed at the customer's site.

(vi) **Delivery:** The delivery of software is done in two phases. In the first phase, the application is deployed on few customers' sites. The purpose of this procedure is to check the product performance at client site, on the basis of feedback from users, whether any changes is necessary or not. In the second phase, finally the product is distributed to the customer's officially.

(vii) **Maintenance:** The maintenance as the set of activities that are performed after the system is delivered to the customer. Its purpose is to monitor the performance of the software, and removing error if nay faced by the customer in site.



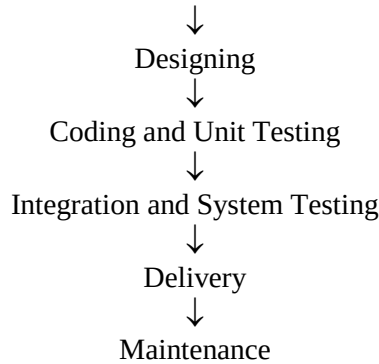


Fig. 1.20 Steps of Waterfall Model

Advantages of Waterfall Model

The various advantages of the waterfall model include:

1. It is a linear model.
2. It is a segmental model
3. It is systematic and sequential.
4. It is a simple one.
5. It has proper documentation.

Disadvantages of Waterfall Model

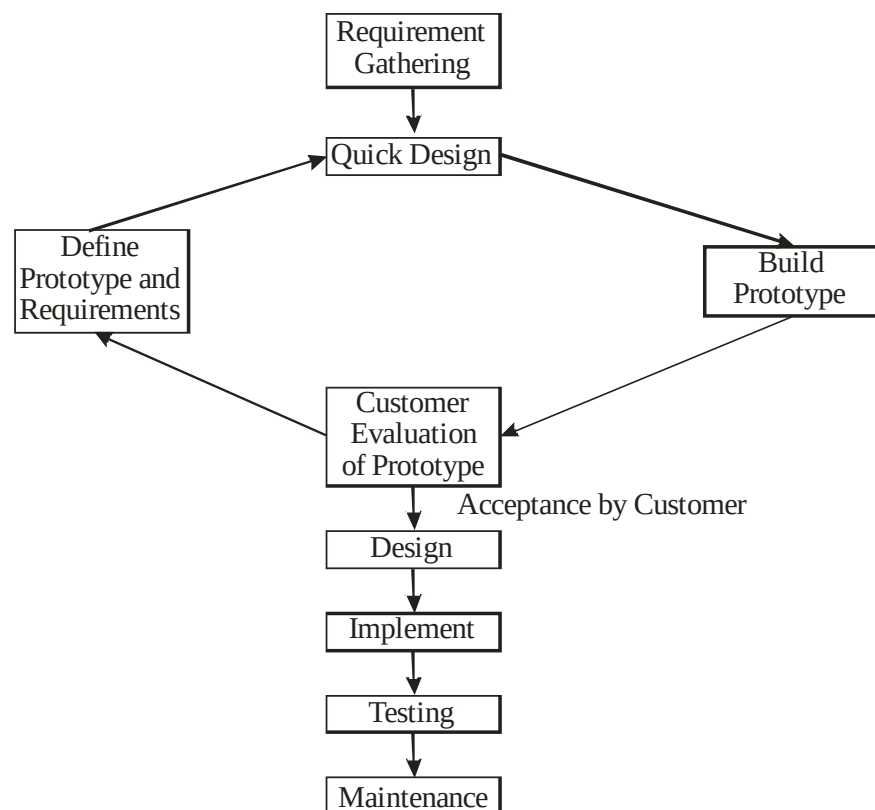
The various disadvantages of the waterfall model include:

1. It is difficult to define all requirements at the beginning of a project.
2. The model is not suitable for accommodating any change in later stage.
3. It does not good for large scale projects.
4. It involves heavy documentations.
5. We cannot go backward in the SDLC.
6. There is no prototyping model for clearly realizing the customer's needs.
7. There is no risk analysis.
8. If there is any mistake in any of seven phases then we cannot develop good software.
9. It is a document driven process that requires formal documents at the end of each phase.

1.19 PROTOTYPING MODEL

1. Prototyping model is an intuitive approach of the waterfall. It overcomes the limitations of waterfall model.
2. Multiple development cycles take place in prototyping, making it multi-waterfall cycle. Cycles are further divided into smaller and easily manageable iterations.
3. The interactions in the prototype enable the developer to better understand what needs to be done and to satisfy users.
4. The prototype model is helpful in understanding the currently known requirements. In the waterfall model, client has to specify his/her requirements before the designing and coding phase. Client's requirements are freezed in the requirement phase. But in the prototyping model, client can specify his/her requirements in more detailed manner.
5. In this model, the client can interact with the prototype and can get an actual feel of the system.
6. Prototyping model is suitable for those projects where it is very difficult to determine the specifications of complex and large projects like client's feedback regarding software product, what is correct and missing, what needs modification etc.

7. Developers make changes in the software product on client's suggestions. Clients use the new changed software product again and allowed to give his/her suggestions to the developers. This cycle repeats until client satisfies with the software product.
8. Prototyping ensures that end users constantly work with the system. Prototyping model is excellent for designing good human computer interface systems.
9. The steps of prototyping model are as follows:
 - (i) **Requirements gathering and analysis:** The model begins with requirement gathering and analysis in detail. In this phase, requirements are gathered and analyzed by the organization.
 - (ii) **Quick Design:** After analysis of requirements, quick design is created. It is a rough design not a detailed design. This quick design helps in developing a prototype and provides an idea of the software product to the user.
 - (iii) **Build Prototype:** With the help of quick design, a prototype is build which represents the working model of the required software product.
 - (iv) **User Evaluation:** A proposed prototype is evaluated by the user. User studies the strengths and weaknesses of this prototype. User provides his/her comments and suggestions like what is to be added or removed etc. to the developer.
 - (v) **Refine Prototype:** If after the evaluation of the prototype, user does not satisfy, then prototype is refined according to the requirements. A new prototyped is developed and being evaluated by the user. After the satisfaction of user, final system is developed.
 - (vi) **Engineer Product:** Once the user accepts the software product is maintained on regular basis.



Prototyping Model

Fig. 1.21 Prototyping Model

Advantages of Prototyping Model:

The various advantages of the prototyping model are as follows:

1. **Quick development:** Developer's approach is to generate software product quickly according to the client's requirements.
2. **Flexible:** Prototyping models are more flexible as compared to waterfall model. Clients can give suggestions to developers for the changes in the software product. In this way, model reduces the communication gap between client and developers.
3. **Easy to test and debug:** Developers can easily test and debug a smaller iteration rather than the whole software product.
4. **Manage risk:** During the small iteration, risky pieces are identified and handled easily.
5. **User Involvement:** Users are actively involved in the software development process.

Disadvantages of Prototyping Model

The various disadvantages of the prototyping model are as follows:

1. Each phase of iteration is rigid and do not overlap each other.
2. **High Cost:** Experienced developers and changing requirements are two main reasons of higher cost in prototyping model.
3. **Lower Quality:** During the development of software product according to prototyping model, the developer's focus is on quick development without thinking about the quality of software product. Development approach is to satisfy client's needs without bothering about the quality of software product.
4. **Compromises in Implementation:** Developer's aim is to get prototyping quickly and to meet client's desires. To achieve this aim, developers make many implementation compromises like inappropriate operating system, an inefficient algorithm etc.
5. **Time Consuming Process:** It is a slow process. The aim of prototyping model is to satisfy the customer. But requirements of customer keeps changing according to external environment. Hence, it is time consuming process.

1.20 ITERATIVE ENHANCEMENT MODEL

1. Iterative enhancement model provides the benefits of both waterfall model and prototyping model.
2. According to the iterative enhancement model, each increment adds some functional capability to the system until the full system is implemented.

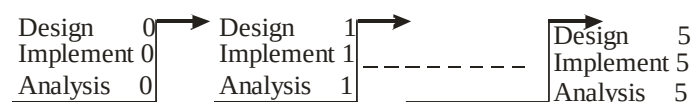


Fig. 1.22 Iterative Enhancement Model

3. The basic idea behind this model is to develop a system through repeated cycles (iterative) and in smaller portions at a time.
4. This model is used when requirements of system are clearly defined and understood.
5. Iterative enhancement model supports changing requirements.
6. Parallel development can be planned and progress can be determined during iterative enhancement model.

Advantages of Iterative Enhancement Model

1. Iterative enhancement model is better in testing. It is because testing increment is very easy as compared to testing the entire system. It overcomes the limitation of waterfall model.
2. Like prototyping, this model also provides feedback to the client. This feedback is very useful for determining the final requirements of the system.

3. This model also provide results periodically and early. It allows the users to get some early experience of the software.
4. Risks are identified and managed during iteration. During smaller iteration, testing and debugging is easy.
5. This model is suitable for large and mission critical projects.

Disadvantages of Iterative Enhancement Model:

1. This model is not suitable for small projects.
2. During iterative enhancement model, management complexity is more and highly skilled resources or required for risk analysis.
3. Sometimes, the iteration may never end and the user may never really get the final product.

1.21 SPIRAL MODEL

1. Spiral model was developed by Barry Bohem in 1988. In this model, software is developed in spiral with many loops.
2. The number of loops in a spiral model are not fixed and can vary from project to project. Each loop is the phase of the software project.
3. At each loop, risks are identified and resolved by prototyping. At every spiral, risk analysis is done to evaluate the development efforts/labour and risks.
4. This model is very flexible because number of phases are not fixed for the software development.
5. Inner cycles/loops represent the requirement analysis alongwith the prototype, whereas outer cycles represent the waterfall model.
6. Generally, spiral model is combination of strengths of various models. It uses stepwise approach of waterfall model and prototyping for resolving the risks occurred during software development.
7. Spiral model is also called as metal model because both waterfall model and prototype model are used in it. It reduce risk as well as follow systematic approach.
8. **Phases of Spiral Model:** There are four phases in the spiral model. In the starting of this model, requirements are gathered to analyze the risk. Spiral model makes communication effective between customer and developer. In this model, angular component represent the progress and reduces the represent cost.

The four phases of spiral model are discussed as follows:

- (i) **Planning:** In the planning phase, requirements are gathered. All methods and strategies are decided in this phase. The objectives, constraints and alternatives are also discussed in this phase.
- (ii) **Risk Analysis:** A process is undertaken to identify the risks. In this phase, alternate solutions are also identified to resolve the risk. Prototyping is used at the end of this phase to resolve the risks which are identified and analyzed. In this phase, risks associated with these new alternative solutions are also analyzed. In this phase, keeping new risks in mind, strong decisions are taken to resolve the risks with prototype.
- (iii) **Engineering:** It involves the actual development of software project. After finding and resolving all the risks through prototyping, software is being developed in the engineering phase.

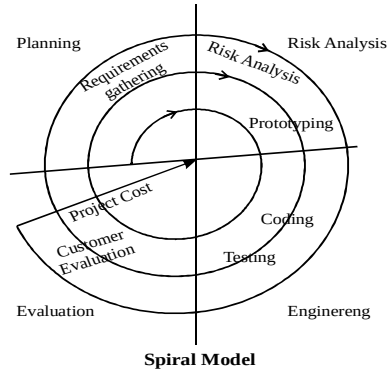


Fig. 1.23 Spiral Model

(iv) **Evaluation:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Advantages of Spiral Model

1. Spiral model evaluates the risk analysis. It enhances the avoidance of risk.
2. It is good for large and mission critical objects.
3. Software is produced early in the software life cycle.
4. Strong approval and documentation control.

Disadvantages of Spiral Model

1. Sometimes, it is costly in use.
2. It is not suitable for small projects.
3. The success of spiral model depends upon the risk analysis phase.
4. Risk analysis requires highly specific expertise.
5. It is not suitable for low risk projects.
6. Spiral may continue indefinitely.

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 2: SOFTWARE PROCESS

2.1 Introduction

2.2 Characteristics of Software Process

2.3 Project Management Process

2.3.1 Definition

2.3.1.1 Characteristics of Project

2.3.1.2 Project Manager Manages the Project

2.3.2 Four P's in Project Management

2.3.3 Activities of Project Management

2.3.4 Issues in Project Management

2.4 Software Configuration Management Process

2.1 INTRODUCTION

Software is developed through a set of activities which are essential for software development of software; a collection of procedure is adopted to achieve certain goals or standards. Therefore the process that deals with technical and management issues of software development in mind is known as software process.

Computer software is loaded into the computer memory and the computer becomes capable of operating the software. The instructions are passed from application software to the hardware through the systems software. The following points are important to be noted about software process:

1. Processes use resources which under terms and conditions product intermediate and final product.
2. Processes are made up of sub processes and each sub process has its own process model.
3. A process has entry as well as exit criteria which controls and monitors the beginning and completion of an activity.
4. A process includes guidelines and it forces uniformity in all activities.
5. Many technical and managerial issues are needed to develop a software.

2.2 CHARACTERISTICS OF SOFTWARE PROCESS

A software process has so many characteristics which have been listed below:

1. **Comprehensiveness:** The process should be easy to understand the user.
2. **Visibility:** The progress of the process should produce clear results.
3. **Robustness:** The process should continue even if there is a unexpected problem.
4. **Supportability:** The process should be such as to support the process activities. It should also make the testing easy and it should be able to produce the software easy to maintain.
5. **Acceptability:** The process should be acceptable to the engineers and it should be usable.

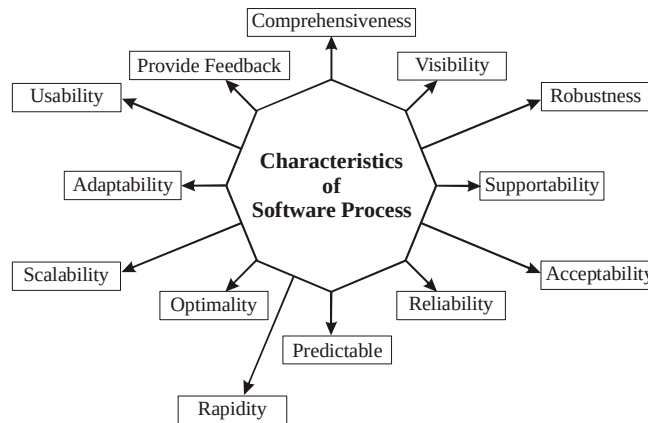


Fig. 2.1 Characteristics of Software Process

6. **Reliability:** The errors in the processes should be avoided or trapped before they result in errors to the product.
7. **Predictable:** The outcome of the project should be predicted before the project is completed.
8. **Rapidity:** The process of delivering the system with given specifications should be rapid.
9. **Optimality:** The process should be able to produce high quality software at a very cheap rate.
10. **Scalability:** It should be applicable for large scale software projects.
11. **Adaptability:** The process should adaptable to the changing requirements.
12. **Usability:** It should have proper user interface and adequate document.
13. **Provide feedback:** There must be feedback information for improvement.

2.3 PROJECT MANAGEMENT PROCESS

2.3.1 Definition

It can be defined as an activity to achieve an objective through the wisely use of knowledge, skills, tools and techniques by the Project Manager.

Project Management is an artistic use of various resources so that the project is completed on time and within budget. It should provide satisfaction to the End user. Project Management is combination of control, Leadership, Team work as a unit, resource management etc.

Process Groups

For the successful completion of a project; a proper system of different activities need to be followed. These processes can be placed into five inter-related process groups: 1. Planning, 2. Organisation, 3. Staffing, 4. Directing, 5. Controlling.

The main job of management is to enable a group of professional to work towards a common goal.

1. **Planning:** This is an important part of project management. The following points are kept in mind by the project management.

- (a) What objectives are to be achieved?
- (b) What resources are needed?
- (c) How and when the resources are to be acquired?
- (d) Ways and means to achieve the goal.
- (e) To determine the flow of information, people and products within the system.

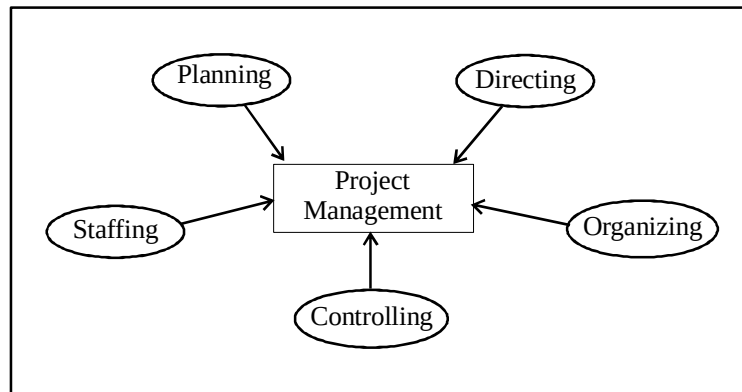
2. **Organizing:** It involves the division of duties and responsibilities among different groups in such a way that the goals of the enterprises are achieved. This is possible only when proper planning is done by the professional within the groups. Best organization means best results in the end.

3. **Staffing:** It deals with hiring of qualified and well trained personnel for the positions that are recognized by the organizational structure. It carries the process of recruiting, hiring, training, rewarding, retaining, compensating, developing and promoting employees.

4. **Directing:** It is the process guiding and leaving the subordinates so that they may understand and identify the organizational structure and the goals to be achieved by the enterprise.

5. **Controlling:** This step includes the process of measuring and correcting activities. The performance of each group is measured against plans and if there is any deviation, then corrective actions are taken to achieve the desired goal.

Software Project Management is based on sound technical skills and Management Skills. The Technical Process of software development depends on the application of sound management principles. It is not possible for one person to handle today's large and complex programs therefore team work is necessary under the leadership of the project manager.



Components of Project Management Process

Fig. 2.2 Components of Project Management Process

2.3.1.1 Characteristics of Project

A project is a temporary endeavor which is taken to provide a unique product or service. A project is different from production work in the sense that all projects have a beginning and an end. Projects are of different size. It can be quite small such as the repairing of a sofa set. Projects can be too big involving thousands of people and investing large sum of money. A project can take place at any level of organisation or even can include the whole organisation. A project can be of small duration (days or weeks) or long duration (to be completed in several months or years).

Projects are of two types:

(i) **Temporary Projects:** Each and every project has a beginning and an end. The project can be closed due to the following reasons:

- (a) Project's objective is achieved.
- (b) When it becomes clear that it is difficult to achieve the objectives in a practical way.
- (c) The project is no longer required now.

(ii) **Unique Projects:** A project becomes unique when it tries to improve upon what was done in the past. Everything is planned and organized to bring a new product with new specifications.

A successful completion of a project demands better planning, well defined organization, technical monitoring, project scheduling, cost, time and effort estimations. The proper measurements and metrics make a project worth its price and qualities. Composition of teams, assigning different tasks, providing resources and establishment of project standards are the major characteristics of a good project.

2.3.1.2 Project Manager Manages the Project

Project Manager

The project manager is the most important individual who uses both the process and set of tools and techniques to define the aims and objectives of the project. The project manager plans the work to achieve

the goals. He leads the project and support teams, monitors progress and so manages everything that the project is completed in a satisfactory way.

The software project manager keeps an eye on day by day operations. He can have the responsibility for multiple projects. The need for a good project manager has risen properly in the industry and commerce. Project manager is the central axis of the project-based organizations because it is establishing itself both as a professional career path and a way of controlling business.

The main responsibility of the project manager is to develop project plans including the work breakdown structure, time estimated, planning resources, training plan, risk management plans and identifies other plans. He uses PERT chart/Gantt chart etc. for scheduling all the tasks. **The main responsibilities of the project manager are:**

1. **Define Goals:** He defines goals, plans and monitor's tasks and resources. He recognizes and find solution to the issues and in addition to it controls costs and budgets.
2. **Leads System Development:** The project manager showed have a good understanding to the project management task and he leads system development on an expert system developer.
3. **Manages and organizes the primary resources** needed to complete the project. He interacts with people linked to the project and mobilizes resources and equipment very well.
4. **Control and co-ordinates every aspect of the project** and motivates the team members to achieve the project objectives.

To conclude we can say that the project manager, plays a vital role in selecting everything and organizes, plans and controls the development process. He communicated with different persons, participants in meetings and gives presentations.

2.3.2 Four P's in Project Management

A successful venture is the one where the project manager successfully builds the right product using the right process for the right project with the involvement of efficient people.

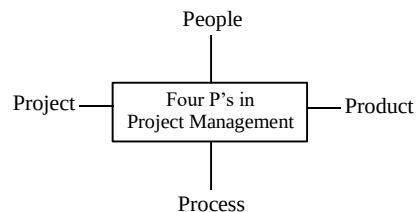


Fig. 2.3 Four P's in Project Management

1. **People:** The skillful and efficient people are the greatest asset of a software organization. The people represent the intellectual capital and efficient managers get best out of their investment in people. Success comes to those organizations which respect the people in the project team. The following people are involved in the software process:

- (i) **Senior Managers:** Defines the business issues.
- (ii) **Project Managers:** Who man, motivate, organize and counting the programmers?
- (iii) **Programmers:** Who deliver the technical skills needed to develop a product?
- (iv) **Customers:** Who specifies the requirements?
- (v) **End users:** Who actually interact with the software once it is developed?

The people who work in a project have to work as a cohesive group and contribute their best to the project. They will have to be organized because the team structure has a direct influence on the product quality and project productivity. The project manager has to solve a lot of problems with the help of the people in his team in the most effective manner. He has to motivate plan and organize the work of the programmers for the proper functioning of the work. The managerial activities of the project manager and his leadership skills have a great influence in building a letter team. He must have the client management skills too. He should be able to communicate effectively with his clients.

2. **Product:** The software product is a tangible item that is produced by a project according to the required specifications. The project manager has to identify opportunities for the use of the product. He must explore the idea about the product in detail. For this scope of the project must be established.

Software Scope: The software product must be built in such a way to fit into a larger system. What constraints are needed must be known in advance. Software project scope must be unambiguous and understandable at the management and technical levels. Quantitative data are stated explicitly, Limitations are noted and mitigating factors are described.

Problem Decomposition: It is an activity that is important for software requirement analysis. Decomposition of the problem is applied in two major areas:

- (i) The functionality which must be delivered
- (ii) The process which will be used to deliver it. Software functions are evaluated and refined to furnish more detail prior to the beginning of estimation. As cost and schedule estimate are functionally oriented; some degree of decomposition is often useful.

3. **Process:** The project manager must decide to select the process model which is most appropriate to be produced by the project team the first requirement is to build the project environment in while the software team works. The attributes of the product must be kept in mind. After the selection of the process model, the preliminary plan is established and process decomposition begins the organization adopts a set of framework activities which include:

- (i) Customer communication
- (ii) Planning
- (iii) Risk analysis
- (iv) Engineering activities
- (v) Documentation and Training
- (vi) Customer evaluation etc.

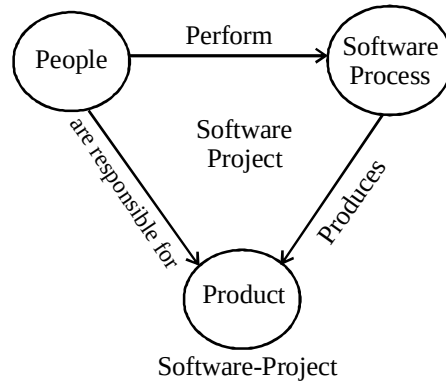
A software team must have the degree of flexibility in selecting the software engineering paradigm which suits well for the project. A relatively small project similar to the past efforts is chosen. If the deadline is very tight then an incremental strategy is adopted. Similarly projects with other characteristics are selected. In the end a more complex project with a broader scope and significant business impact is adopted.

4. **Project:** The software project management activity is just like an umbrella activity within the software engineering process. It begins before the start of any technical activity and continues throughout the definition, development and support of computer software. The project management activities include:

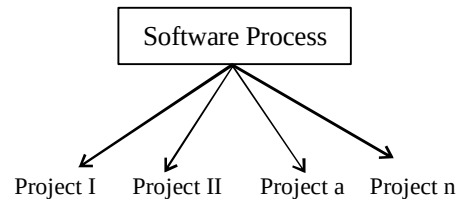
- (i) Measurement and Metrics
- (ii) Management activities
- (iii) Project planning
- (iv) Scheduling
- (v) Tracking
- (vi) Risk management activity

To avoid problems in the project activities, the project team follows the five past common-sense approach i.e. (i) Start on the right tool, (ii) Maintain Momentum, (iii) Track process, (iv) Mask smart data.

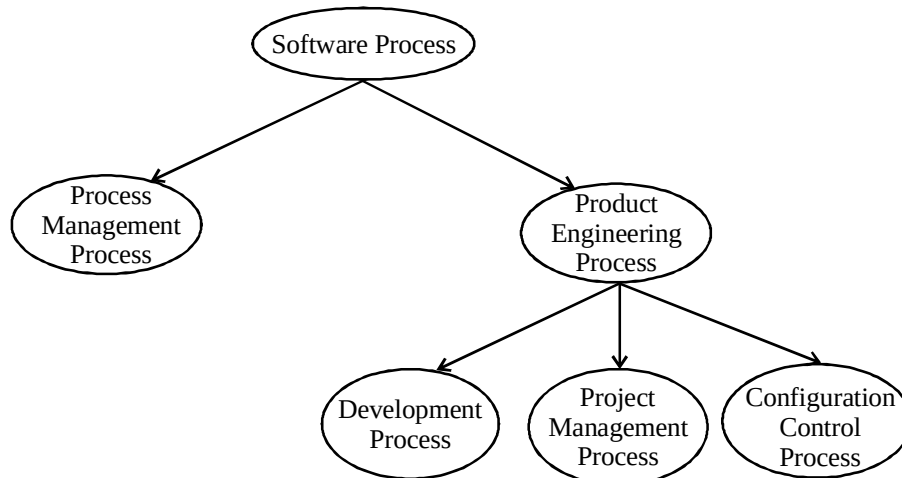
The inter relationship between 4 P's is best illustrated with the diagrams as under:



(A) Inter-Relationship between 4 P's



(B) Inter-Relationship between Process & Project



(C) Process Categories

2.3.3 Activities of Project Management

The following are the major activities which are to be performed by the Project Manager for successful management of the software project:

1. **Proposal Writing:** These are no set rules or guidelines for proposal writing. It only depends on the professional skills and experience proposals describe the aims and objectives of the project and the manner in which it will be carried out. If the proposal writing is effective; only then it will be accepted and contract will be awarded.

2. **Project Planning:** It is just like a Guide Map which guides the development team towards the achievement of goals. Project plan keeps the following points in mind.

- Identifying different activities
- Setting of Goals
- Identifying the final result in the form of deliverables to be produced by the project.

3. **Project Scheduling:** It involves the following things

- Breaking the main project into smaller tasks and identifying activities
 - Estimating resources for these activities
 - Judging the time required to complete these activities
 - Allocation of people to different activities and coordinating their tasks.
 - Full utilization of human resources
4. **Project Cost Estimation:** This activity is concerned with guessing and calculating the resources required completing the project plan.
5. **Project Monitoring:** In this activity the project manager check the progress of the project from time to time and compares the actual and planned progress on costs. If there is any deviation then project manager takes corrective steps.
6. **Project Review:** This activity is concerned with observing the overall progress and technical development and to study whether the project's status is in accordance with the aims and objectives of the organization commissioning the software.
7. **Personnel Selection and Revaluation:** The project manager selects a balanced team of experienced employees and youthful fresher's. It is necessary that the services of the staff are utilized effectively.
- For selecting personals, a project manager has to keep in mind the following steps:
- Project Budget
 - Impossible to appoint new staff
 - Providing Training to employees
8. **Project Report Writing:** A Project Manager prepares Project Report. He is responsible to submit this report to both the client and contractor organization. He must have the written as well as oral skills to present his report in a very effective way. The language of the project report should be very simple and clear to understand.

2.3.4 Issues in Project Management

There are so many issues which are to be kept in mind in the Software Project management. There are specific goals that are to be achieved. Certain Laws and Principles have to be followed. The difficulties and challenges which stand in the way of management have to be overcome.

Goals

1. To develop the software within the specified budget and within the planned resources. It should be economically justifiable.
2. Time schedule is of utmost importance. The software should be produced on time and is ensured to deliver on time.
3. To develop the software according to the parameters and requirements of the user.

Principles

The following are the principles that are to be followed:

1. Before starting the project, aims and objectives should be clearly understood.
2. The project manager should understand various constraints which may be of hardware, software, cost or any other type of constraints.
3. Making plans in advance to achieve the goals in a better way. Cost estimation, Need for human resources and other factors should be planned well.
4. The project should be monitored so that no problem goes unnoticed.
5. Adjustments, modifications and appropriate changes as and when necessary should be made for appropriate result.
6. A healthy and good work environment should be provided so that each member gives his/her best performance.

Difficulties and Key Challenges

Software project are unmatchable and developed.

1. **Redundancy Problem:** Software projects are unmatchable and developed only once. They face redundancy problem in some cases.
2. **Multiple Solutions:** It becomes very difficult to decide which way to go when there are multiple solutions for a given problem.
3. **Difficult Cost Prediction:** It is difficult to predict personnel cost because they have different capabilities.
4. Technological advancement in hardware and software technologies creates a big obstacle in the development of the software.
5. Software is very difficult to monitor due to its intangibility.
6. Changes in software specifications and design can take place during any phase of software development.
7. **Not Well Designed:** The responsibilities of different team members are not sometime well defined. The user requirements are not precisely defined.

All these difficulties need to be overcome so that the software is developed within the specified time and estimated costs and fulfills the user requirements.

2.4 SOFTWARE CONFIGURATION MANAGEMENT PROCESS

Introduction

Software configuration management or in short SCM is the discipline of identifying, defining and controlling changes throughout the life cycle. It includes revision control and the establishment of baselines. Sometimes it so happens that changes arise due to near business or market conditions and the atmosphere so created requires a change in the final product. New user requirements also need modifications of data or functions produced by the software. Other reasons may be business growth, downsizing, cost and time constraints. SCM manages changes in the intermediate products in a controlled manner so as to maintain the integrity of the product. The traceability is also maintained and the development of the product is made more manageable.

Software design may change at different points of time like after testing phase or coding phase. The main points or goals of SCM are as follows:

1. **Configuration Identification:** The structure of the product is identified and then its components and their types are also identified. CCM then makes these products unique and accessible in some form.
2. **Control:** Implementing a controlled change process. For this a change control board is set up which approves or rejects all change requests. It answers consistent software by the creation of a baseline product.
3. **Configuration Status Accounting:** Recording and reporting the status of the components and change requests. It collects all information and vital statistics about the status of the development process.
4. **Auditing and Reviewing:** Validating that the configurations contain all the intended parts are sound. It keeps consistency with respect to specifying documents and ensuring that the product is a well-defined collection of requirements, architectural specifications and user manuals.
5. Build management, process management and environment management all ensure adherence to the organization's development process.
6. Facilitating team work and making sure that every defect has been tracked back to the source.

Software Configuration Items

It is one of the smallest unit of change. IEEE defines it as "an aggregation of hardware, software or both that is designated for configuration management and treated as a single entity in the configuration management process".

Base lines

Whenever there is a need for a change, a basis for the change must be clearly known to making changes. A baseline helps to control change. IEEE defines a baseline as a specification or product that has been formally renewed and agreed upon that thereafter serves as the basis for further development and that can be changed only through formal change control procedure.”

Activities of SCM

The various activities involved in configuration management are:

- **Configuration identification:** The structure of the product is identified and then its components and their types are also identified. CCM then makes these products unique and accessible in some form.
- **Change control:** Implementing a controlled change process. For this a change control board is set up which approves or rejects all change requests. It answers consistent software by the creation of a baseline product.
- **Configuration audit:** Validating that the configurations contain all the intended parts are sound. It keeps consistency with respect to specifying documents and ensuring that the product is a well -defined collection of requirements, architectural specifications and user manuals.
- **Configuration status reporting:** All these activities ensure that all the changes are properly implemented.

Changes in Software Development

— Diagrammatic Illustration.

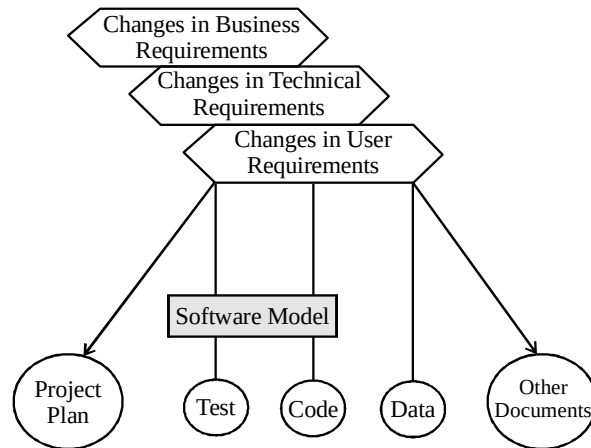


Fig. 2.4: Diagrammatic Illustration of Changes in Software Development

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 3: PROJECT PLANNING

3.1 Software Project Planning

3.1.1 Main Objectives

3.1.2 Principles of Project Planning

3.1.3 Components of Project Planning

3.1.4 Activities in Project Planning

3.2 Software Project management Planning

3.3 Software Cost Estimation

3.4 Software Sizing

3.4.1 Approaches/Types of Sizing

3.4.1.1 Lines of Code

3.4.1.2 Function Point

3.4.1.3 “Fuzzy Logic”

3.4.1.4 Change Sizing

3.5 Problem Based Estimation

3.6 Process Based Estimation

3.7 Cocomo-Model (Constructive Cost estimation Model)

3.8 The Software Equation

3.9 Project Scheduling

3.10 Basics of Software Measurements

3.11 Types of Software Measurement

3.12 Criteria of Measurement

3.13 Process of Measurement

3.14 Basic Concepts of Software Metrics

3.15 Importance of Software Metrics

3.16 Categories/Types of Software Metrics

3.16.1 Projects Based Metrics

3.16.1.1 Product Metrics

3.16.1.2 Process Metrics

3.16.1.3 Resource Metric

3.16.2 Design Based Metrics

3.16.2.1 Traditional Metrics

3.16.2.1.1 McCabeCyclomatic Complexity (CC)

3.16.2.1.2 Source Lines of Code

3.16.2.2 Object Oriented Metrics

3.16.2.2.1 CK (Chidamber&Kemerer's Metrics Suite)

3.16.2.2.2 Mood (Metrics for Object Oriented Design)

3.17 Software Metric Approach

3.18 Guidelines for Metrics

3.19 Qualities of Good Software Metrics

3.20 Software Quality

3.21 Attributes of Quality Software

3.22 Software Quality Management System

3.22. 1 Software Quality Management Principles

3.23 Software Quality Assurance (SQA)

3.23.1 Importance of Software Quality Assurance (SQA)

3.23.2 Activities of Software Quality Assurance(SQA)

3.23.3 Software Quality Metrics

3.23.4 Principles of SQA

3.23.5 Software Quality Assurance Standards

3.24 Software Quality Control (SQC)

3.25 Difference between quality assurance and quality control

3.1 SOFTWARE PROJECT PLANNING

Project Planning is the most important aspect of software project management. Project planning prepares a list of activities that need to be done to achieve the goal. The main purpose of project planning is to clarify the goals, needs and constraints. The success of a project depends upon its planning because poor planning is a sure ticket to failure and poor quality result with high maintenance cost. A good planning prepares a road map in advance what to do, how to do, when to do, who is to do and how much will it cost to complete the project.

3.1.1 Main Objectives

Project Planning is a well-organized and an integrated process which keeps in mind the ultimate goal to achieve:

- Project Planning aims at better utilization of resources.
- To finish the project in time.
- Defines roles of different team members.
- Assigns duties and responsibilities.
- It determines project constraints.

A large number of people are involved in the project planning. These include

- (i) Senior Management which is responsible for employing personnel's and providing resources.
- (ii) The project management team which includes project managers and developers. They are responsible for planning determining and tracking activities, selecting all ways and means to achieve the goal. The project manager is the key to successful project planning.

3.1.2 Principles of Project Planning

The following principles or fundamentals of software project planning are to be followed:

- **Planning is the basic need:** The software project management process begins with a group of activities called Project Planning. This is one of the most important management activity. It is necessary for modern business because helps in better co-ordination and reduces/removes uncertain. Planning is a never-ending process.
- **Risk Analysis:** It is the duty of the Senior Management and Project Management to study all the risks and hindrances that may team affect the final output.
- **Tracking of Project Plan:** The project plan should be continuously tracked and modified as per the requirements.
- **Quality Products:** Those process should be identified which can ensure quality in software.
- **Description of Flexibility:** Planning should be flexible so as to accommodate new changes the project is in progress.

3.1.3 Components of Project Planning

Different types of projects can have different components depending upon their working and implementation. Here are some common and powerful components of project plan.

1. **Project Scope:** Each project has its own wide project scope.
2. **Project Schedule:** Every project has its own time frame within which it must be completed.
3. **Project team organization:** Each project has a well-organized team under a project leader.
4. **Technical Description:** Each project has a well-organized team under a project leader.
5. **Project Standards, Procedures and proposed technical tools:** For success, completion of the project, it must have all the standards, process and technical tools.
6. **Quality Assurance Plan:** Each project has the assured plan for successful completion of the project.
7. Special Development Tools and Techniques are needed.
8. **Configuration Management Plan:** For the long range planned projects.
9. **Documentation Plan:** A project must have the documentation which is well-planned for the maintenance of the project at the later stage.
10. **Data Management Plan:** There should be Data Management Plan for every working project in the short range planning.
11. **Resource Management Plan:** There must be a Resource Management Plan to implement the project in a successful manners.
12. **Test Plan:** A detailed and ordered test plan is needed for the successful testing of two project.
13. **Training Plan:** It must be set for the users for the successful completion of the project and working of the project.

14. **Security Plan:** There should not be any un-authorized access. For this security plans like the secret purpose password should be established.
15. **Risk Management Plan:** A Project Manager must consider all the constraints and possibilities to avoid and type of risk management.
16. **Maintenance Plan:** Maintenance procedures and plans are needed to avoid any type of threat during working hours.

Planning is done on both formal and informal basis. The main aim is to provide the guidelines for decision making are based on future expectations.

3.1.4 Activities in Project Planning

Project Planning has various aspects like estimating, scheduling and assigning the project resources in such a way to deliver the final product of desired suitable quality. However activities have an important role which can determine the success of the project. The various planning activities team members will need to do are given as under:

- The recruitment of the team and building of the team as a unit.
- To organize the project
- To know and confirm the start and end dates by preparing a project schedule.
- To create the project budget.
- To know about the customer requirements for the final result.
- Limiting the project scope boundaries and defining them properly.
- To write a description of the final result.
- To assign duties
- To assign accountability.

A typical project plan can include the following activities which have been briefly discussed as below:

1. **Project Scope Definition and Scope Planning:** The assumptions, constraints and user expectations along with business needs, technical needs and project deliverables etc are documented. Project objectives and any other thing that defines the final product requirements is also documented.
2. **Quality Planning:** This is an important part of project planning. Quality of the final product can't be over looked. For this various factors that influence the quality of the final product are taken into account.
3. **Project Activity:** Definition and activity sequencing: All activities are defined in advance which are required to deliver the product by producing the various product deliverables.
4. **Time, Effort and Estimation of Resources:** Effort, time and resources required are clearly estimated and documented in this step. Many techniques like function points, lines of code, complexity of code, Bench mark etc. can be used to calculate effort.
5. **Risk Factor Identification:** One should be ready to face the unexpected things in life. It is very important to pin point the risk factor based on the assumptions, constraints, user expectations and challenging conditions.
6. **Schedule Development:** Project Scheduling is one of the main task of project planning. Time frame work is to be prepared for each of the inter-dependent activities. It will have an influence on the cost estimates and the cost benefit analysis.
7. **Risk Management Planning:** In this process risk factors are identified and analyzed. A risk resolution plan is prepared to minimize the impact of the risk factor on the project.
8. **Cost Estimation and Budgeting:** The costs involved in the execution and implement of the project is estimated and only then budget allocation is done for different phases of the project.
9. **Organizational & Resource Planning:** The main objective of resource planning is the smooth and efficient running of the project. There are many types of resources like equipment, personnel, facilities, money etc. These resources will have to be fully utilized to achieve the goal.

10. **Project Plan Development & Execution:** The information gathered from all planning processes is used and each of the project tasks and activities are monitored and controlled from time to time delays are analyzed and the project plan is adjusted accordingly.

11. **Performance Reporting:** The progress of each of the tasks is compared with the schedule. Various Techniques such as EVM (Earned Value Management) can be used to measure the performance. Tools like PERT Charts, GANTT Charts, Logical bar charts, Histograms, Pie charts etc. can be used to report the performance.

12. **Planning Change Management:** If the need arises then certain aspects of the project can be changed, but the changes should not have any negative effect on the environmental or the performance. Changes should be studied properly and only then the project plan may be modified.

13. **Project Rollout Planning:** The success of the project depends on the project rollout and implementations. Where a project is rolled out; it has greater impact on the entire technical systems and the business systems. The users should be read to accept it and use it effectively. The users need to be trained to adapt to the new system.

3.2 SOFTWARE PROJECT MANAGEMENT PLANNING

This is an important part of project management. The following points are kept in mind by the project management.

- (a) What objectives are to be achieved?
- (b) What resources are needed?
- (c) How and when the resources are to be acquired?
- (d) Ways and means to achieve the goal.
- (e) To determine the flow of information, people and products within the system.

Organizing

It involves the division of duties and responsibilities among different groups in such a way that the goals of the enterprises are achieved. This is possible only when proper planning is done by the professional within the groups. Best organization means best results in the end.

Staffing

It deals with hiring of qualified and well trained personnel for the positions that are recognized by the organizational structure. It carries the process of recruiting, hiring, training, rewarding, retaining, compensating, developing and promoting employees.

Directing

It is the process guiding and leaving the subordinates so that they may understand and identify the organizational structure and the goals to be achieved by the enterprise.

Controlling

This step includes the process of measuring and correcting activities. The performance of each group is measured against plans and if there is any deviation, then corrective actions are taken to achieve the desired goal.

Software Project Management is based on sound technical skills and Management Skills. The Technical Process of software development depends on the application of sound management principles. It is not possible for one person to handle today's large and complex programs therefore team work is necessary under the leadership of the project manager.

3.3 SOFTWARE COST ESTIMATION

Cost estimation is the most important task of a project. An accurate and scientific estimation of cost, effective usage of resources and time schedule is the basis for a successful completion of a project. Cost

over-run may enrage. The customer and it can lead to the cancellation of the project. There are some factors such as complexity of the project, time availability, reliability and software size which mean a lot of cost estimation. It can be done during any phase of software development. It depends upon software information regarding user requirements, design and source code etc. It becomes easier to estimate cost in the later stages as more information becomes available. To avoid unforeseen delays and risks; cost estimation should be done throughout the life cycle of the project. The following diagram accurately depicts how the cost estimation becomes more and more accurate as additional software information is pouring in.

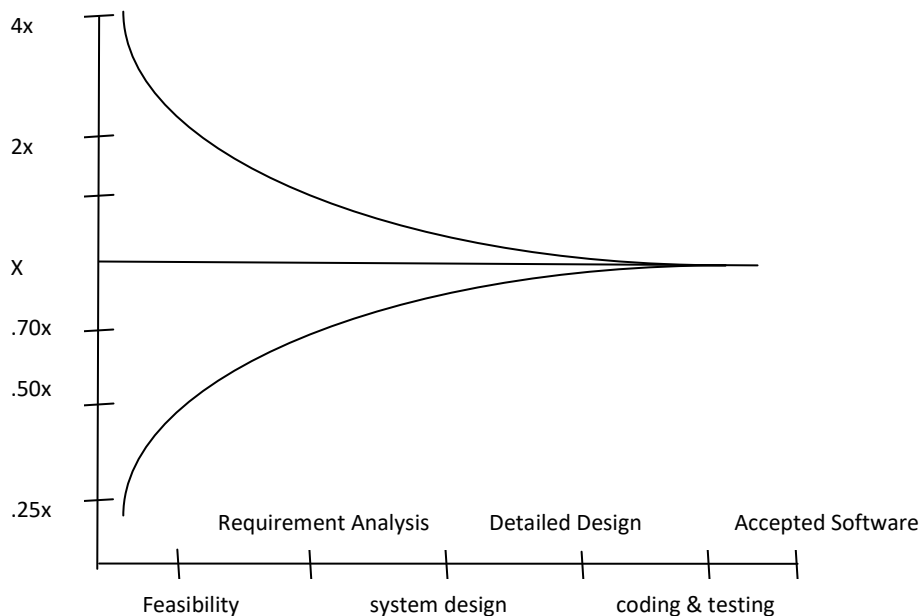


Fig. 3.1: Accuracy of Cost Estimation

The diagram clearly shows that as we proceed from requirement phase to coding and testing phase via system design phase; cost estimates becomes more accurate.

Cost Factors

The following factors influence the cost estimation:

1. **Experience:** If the software developer is well trained in programming language, operating system and hardware the cost will be less.
2. **Product Complexity:** The cost of software project rises up with the level of complexity. Three levels of product complexity are organic, semi-detached and embedded programs.
3. **Project Size:** A large sized project is more costly as compared to a small sized project. More effort is required for a large sized project.
4. **Available Time:** If development time is decreased, then the software project requires more resources and effort and thus the cost increases.
5. **Programmer Ability:** Efficient and capable software programmers bring down the expenditure whereas inefficient programmer increases the cost. Therefore, it is necessary that a programmer should be well-trained.
6. **Level of Technological Know:** How also has a great influence on cost estimation? Modern practices, System analysis, Design techniques etc. have a great effect on cost estimation.

7. **Reliability:** Reliability of the software project depends on the level of accuracy, robustness and consistency etc. cost estimates depend on the level of analysis, design, implementation and verification effort.

The cost estimation should be well-planned. It should be reviewed at regular intervals. The software process should be continually updated and improved upon.

3.4 SOFTWARE SIZING

Sometimes a problem is too big to be understood in a single attempt, similarly software cost estimation is not simple as it appears to be a complex problem like software cost estimation is broken up into fragments to achieve an accurate cost estimate.

Mainly two approaches are adopted:

- (i) Problem based estimation
- (ii) Process based estimation

A project planner must establish the estimate of software size which is considered to be the Quantitative result of the software project. The software is divided into smaller parts because it is convenient to calculate and know about the size of the smaller components. They are less complex. The addition of these smaller components gives an idea of the overall estimate of software size.

Sizing Approaches: There are two types of approaches for estimating size:

1. **The Direct Approach:** Measures sizes in terms of (LOC) lines of code.
2. **The Indirect Approach:** Measures in terms of functional point (FP).

Parameters: The following parameters influence the accuracy of size estimates:

1. The degree of properly estimating the size of the software.
2. The capability of converting Size Estimate to human effort, calendar time and money.
3. The ability of the software team.
4. The stability of product Requirements and the overall environment supporting the development process.

3.4.1 Approaches/Types of Sizing

The project planner finds that, the estimation of size is very important in determining the cost estimation. This problem is also known as Software Sizing. To solve this problem, the following methods can be used:

1. **Fuzzy Logic Sizing:** The planner identifies the application type and its magnitude on a quantitative scale. The magnitude is then refined within the original range.
2. **Function Point Sizing:** In this method the functionality produced by the software system is measured the function point estimates.
3. **Standard Component Sizing:** Standard components can be in the form of modules, screens, reports, lines of code and soon. The number of time a component occurs is estimated and the historical data helps to find out the delivered size of each standard component.
4. **Change Sizing:** Through this method, already existing project is modified and the number and type of the modifications are estimated.

3.4.1.1 Lines of Code

It can be defined as the number of delivered lines of code. Lines of code are a measure to calculate productivity metrics. LOC is used as follows:

- (i) As an estimate variable to size each and every element of the software.
- (ii) It is used as baseline metric obtained from the previous projects and used with estimation variables to develop projections for cost and effort.

Line of code is the most commonly used software size metrics. LOC depends on the programming language. In assembly language lines of code will be higher as compared to lines of code in high level language like C++/Java.

In the earlier stages of development, less information becomes available whereas full information becomes available only at the end of the project. So exact number of lines of code can be determined after the completion of the project.

Guidelines

1. One line of code is for One Logical line of code.
2. Lines of code delivered as a part of software are included.
3. Lines of code for drivers, test stubs and other support software are excluded.
4. Software code by software developer is included whereas the code created by the application generators are excluded.
5. Declarations in the programme are counted as LOC whereas comments in the programme are not counted as LOC.

Keeping in view the historical Data, The project planner estimates. Three values optimistic S_{opt} , most likely (S_m) and pessimistic (S_{pess}) for each size.

The equation ($S_{opt} + 4 \times S_m + S_{pess}$) is used to compute expected value. The following example of a software will clarify how the estimating size is determined. Here in the example software is a combination of six functions — (i) user face, (ii) word processing, (iii) file storage and retrievals, (iv) Data base management, (v) Word processor and (vi) Peripheral control.

Estimating Size

Function	S_{pess}	S_m	S_{opt}	Expected Size $S = \frac{1}{6} (S_{opt} + 4 \times S_m + S_{pess})$
User interface	1500	1800	2100	1800
Word Processing	1800	2400	3000	2400
File Storage & Retrievals	1700	2100	2400	2083
Data base management	2500	3000	4000	3083
Word Processor	1200	1600	2000	1600
Peripheral Control	1300	1700	2000	1683
Total Expected Lines of Code =				12649

Table showing method of calculating Size Estimation

The developer estimates the size of each and every function:

S for user interface =

S for word processing = and soon

We see that size of the software in terms of

LOC is = 12649

3.4.1.2 Function Point

Function Point lines of code are the most important methods for size estimation. The function point metric was devised and suggested by A.J. Albrecht. This is used to measure functionality delivered by the system; Function point estimates in determining estimate effort required to design, code and test software, predict the number of errors and it so forecasts the number of components being used in the system.

The measures of software information domain value and software complexity are the two things needed to find out function point through an empirical relationship.

Software complexity is of three categories; of simple, average and complex level.

Information domain value is a combination of the following points.

1. **Number of external inputs or (EI):** Users and other applications are a source of providing application oriented data.

2. **Number of external outputs or (EO):** These refer to Reports, Screens, Error, messages etc. The user gets information from each external output. Individual data views are not counted separately.

3. **Number of external inquiries (EQ):** Online inputs are referred to as external inquiries. These in turn. Generate responses in the form of online outputs. In this case each and every distinct inquiry matters and is counted separately.

4. **Number of internal logical files/(ILF):** Logical grouping of data present in the application boundary e.g. master file as part of database is known as internal logical file.

5. **Number of external interface files (EIF):** Logical grouping of data which is external to the application e.g. Data files on tape or disk is known as external interface file.

The complexity value for each count is determined after gathering information about information domain value function point method helps in determining about a particular entry whether it is simple, average or complex.

Estimating Functional Points: Functional points in software are estimated by the equation as given below.

$$F.P = \text{Count Total} (0.65 + 0.01 \times \sum (f_i))$$

Count Total = Sum of all F.P. entries

f_i (i = 1 to 14) are value adjustment factors.

The value adjustment factors are based on the answers to 14 Questions. These questions are answered using a scale of 0 to 5 where

- 0- means – No influence
- 1- means – incidental
- 2- means – Moderate
- 3- means – Average
- 4- Significant
- 5- Essential

Weighting Factor Table 1

Measurement Parameter	Count		Simple	Average	Complex		F.D. Count
Number of User inputs	21	×	3	5	7	21×5=	105
Number of user output	15	×	2	4	6	15×4=	60
Number of user inquiries	25	×	4	5	6	25×5=	125
Number of files	7	×	8	10	14	7×10=	70
Number of external interfaces	3	×	5	7	10	3×7=	21
Count Total							381

Value Adjustment Factors Table II

Sr. No.	Factors	Disadvantages
1.	Backup and Recovery	2
2.	Data Communications	2
3.	Distributes Processing	1
4.	Performance Critical	3
5.	Existing operating environment	2
6.	Online data entry	3
7.	Input transactions over multiple screens	4
8.	Master files updated online	3
9.	Information domain value complex	4
10.	Internal processing complex	2
11.	Code design for reuse	3
12.	Conversions/installation in design	3
13.	Multiple installations	4
14.	Application design for change	4
	Value adjustment factor	40

Computations for Function point: To estimate size in terms of function point table I and Table II are used as follows

FP count = Count X weighting factor (AV)
 From Table I FP for number of used inputs = $22 \times 4 = 88$
 using Table I and II F.P. count Total $\times [0.65 + 0.01 \times ?fi]$
 $= 381 \times [0.65 + 0.01 \times 40]$
 $= 381 \times [0.65 + 0.40]$
 $= 381 \times 1.05$
 $= 400.05$
 $= 400$

3.4.1.3 “Fuzzy Logic”

A project estimate is only as good as the estimate of the size of the work to be accomplished. Different approaches to software sizing are given as under:

1. Lines of codes
2. Functions point
3. Fuzzy logic
4. Change sizing

Fuzzy Logic sizing is a type of approach in which the project planner must identify the application type and its magnitude on a quantitative scale. After this the magnitude is refined which in the original range.

“Fuzzy Logic” approach applies the approximate reasoning techniques that are the corner stone of fuzzy logic. The personal experience can find a place but the planner should also keep on eye on the historical database of the projects so that estimates can be compared to original/actual experience.

3.4.1.4 Change Sizing

Sizing signifies the project planner’s first major challenge. Sizing steps are essential to make valid comparison cross systems or within systems. Productivity can be computed with the help of a software sizing measure.

The change sizing approach is followed only when there arises a need for changes in the already existing project specifications. This modification is used in the new project to bring a required change in its accomplishment. The planned estimates, the number and type (e.g. reuse, adding code, changing code, deleting code) of modifications that must be finalised. By the application of an “Effort Ratio” for each

type of change, the size of the change may be guessed and this new estimate will help to produce the software according to the user requirement.

3.5 PROBLEM BASED ESTIMATION

LOC and F.P. data are used in two ways:

- As an estimation variable to 'size' each element
- As baseline metrics gathered from past projects to develop cost and effort projections.

Although LOC and F.P are clearly different techniques but they have many common characteristics. After a bounded statement about software scope, the project planner decomposes the software into problem functions so that they may be estimated individually. LOC or FP is then estimated for each function.

Using baseline productivity metrics, proper estimation variable and cost or effort for the function is calculated.

LOC and FP estimation are different so far as their technique of decomposition and the target of the partitioning is concerned when LOC is used the decomposition approach supposes that all functions can be broken up into sub functions. LOC estimates depend on the degree of partitioning. For FP estimates decomposition works in a different manner. Instead of focusing on function, it estimates on the basis of information domain characteristics i.e. inputs, outputs, data files, inquiries and external inferences. In addition to 14 complexity adjustment values are estimated and a FP value is derived that can be linked to past data and this is used to generate an estimate.

3.6 PROCESS BASED ESTIMATION

In order to achieve objectives in software project developed a specified and certain process is followed. The estimation of effort in a software is to base the estimate on the process to be followed. The whole process is divided into smaller tasks and after the identification of these tasks; the effort for each task is estimated. These smaller tasks are such as analysis, design and coding etc. These tasks are ordered according to the specified sequence. The project managers take the product size as a major productivity indicator and they become capable of tracking a project. A number of activities are performed in process based estimation. These activities can be listed as follows:

1. Customer communication
2. Planning
3. Risk Analysis
4. Engineering (analysis/Design)
5. Construction release and so on

It is the duty of the project planner to combine the functions and activities to calculate the effort required for each function. The labour rates which vary from task to task are applied. Top level management activities are more expensive on compared to activities performed by the junior staff in the beginning of the framework activities. The accuracy in process based estimation depends on many parameters.

- The degree of sizing of the software and the proper estimation.
- Changing size estimate into human effort, calendar time and money.
- Ability of a software team
- The stability of product requirements
- Environment supporting the development process.

The software considered for process based estimation is divided into functions and for each function a set of tasks and activities are taken last of all, the effort for each function and activity is calculated.

Activity	Customer Communication	Planning	Risks Analysis	Engineering		Construction Release		CE	Total
Task				Analysis	Design	Code	Test		
Function 1				0.50	2.00	0.40	5.00	n/a	7.90
Function 2				0.70	4.00	0.60	2.50	n/a	7.80
Function 3				0.50	3.00	1.00	3.50	n/a	8.00
Function 4				0.50	3.00	0.75	2.50	n/a	6.75
Function 5				0.50	2.00	1.00	2.50	n/a	6.00
Function 6				0.25	1.00	0.50	2.50	n/a	4.25
Function 7				0.50	1.00	0.50	1.00	n/a	3.00
Total	0.25	0.25	0.25	3.45	16.00	4.75	19.50	-	43.70
%Effort	1%	1%	1%	7.66% = 8%	35%	11%	43%	-	

Table showing percentage effort for each function & Activity

3.7 COCOMO-MODEL (CONSTRUCTIVE COST ESTIMATION MODEL)

Barry Boehm gave birth to the idea of COCOMO Model in early 19805 to estimate total effort required to develop a software project. This Model is widely used and if is based on the observation of previously developed software projects. According to Boehm any software development project can be categorized into three type's organic, semidetached and embedded depending on the development complexity.

Organic projects: These projects are in an area in which the organization has enough experience. These project are small in size (not more, the 50 KOLOC). Such system has small teams with prior experience. The term members work together with through understanding and co-operate to accomplish user requirements which are less stringent examples: Simple business system, simple inventory management system etc.

Semi-detached: In such a project, the development team consists of a mixture of experienced and inexperienced staff. The size of such projects is not more than 300 KDLOC

Examples: Operating system, compiler design and data base design

Embedded Projects: These projects are ambitions and novel, they have size more than 300 KDLOC. The organization has little experience developers have to achieve stringent user requirements and these systems have tight constraints from the environment (hardware, software and people)

Example: Software systems used in avionics and military hardware, real-time command system.

COCOMO model estimates the total effort in terms of person-months of the technical project staff. the efforts estimate includes development, management and support tasks but does not include the cost of the secretarial and other staff. Our person month means the effort an individual can typically put in a month. The basic steps in this model are:

1. Obtain an initial estimate of the development effort from the estimate of thousands of delivered lines of source code. (KOLOC)
2. Determine a set of 15 multiplying factors from the different characteristics of the project.
3. Multiply the initial estimate (E_i) with all the multiplying factors.

The initial estimate E_i in person\^-months is determined by the equation $E_i = a * (KDLOC)^b$. The value of the constants a and b depend on the type of the project. KDLOC is used as the measure of the size. COCOMO is based on the hierarchy of three models: basic model, intermediate model and advance model.

Basis Model: In this model only the size of the project is taken in given an approximate of the project parameters.

The Expression for calculating effort is as follows

$$\text{Effort} = a * (KLOC)^b \text{ PM}$$

Basic Model			
Project Type	Constant (a)	Constant 'b'	E
Organic	2.4	1.05	$2.4 \times (30)^{1.05}$ for 30 KDLOC = 85 pm
Semi-detached	3.0	1.12	$3.0 \times (30)^{1.12}$ for 30 KDLOC
Embedded	3.6	1.20	$3.6 \times (30)^{1.20}$ for 30 KDLOC

Effort Calculation

By using logarithmic tables; effort can be found out. Estimation of development time (T_{dev}): The formulator estimating the development time based on the effort are as follows:

Organic : $T_{dev} = 2.5 (\text{effort})^{0.38}$ months
Semidetached: $T_{dev} = 2.5 (\text{effort})^{0.35}$ months
Embedded : $T_{dev} = 2.5 (\text{effort})^{0.32}$ months

e.g. if an organic type software product is estimated to 32000 lines of source code and a software engineer gets Rs. 6000 as salary par month; then nominal effort, nominal development time and cost required to develop the project an be calculated as

$$\text{Effort : } 2.4 (3.2)^{1.05} = 91 \text{ PM}$$

$$T_{dev} : 2.5 (91)^{0.38} = 14 \text{ months}$$

$$\text{Cost: } 14 \times 16000 = 224000$$

Intermediate COCOMO Model

This model takes other relevant parameters into account besides development time and effort the intermediate COCOMO model refines the initial estimate (E_i) by using a set of 15 multipliers based on different characteristics of the software development to calculate total effort in this model, the procedure is as follows:

1. Calculate (E_i) considering the size in terms of KDLOC.
2. Identify 15 parameters and each parameter is given a numerical value called multiplying factor.
3. Effort adjustment factor (EAF) is derived by multiplying all multiplying factors with each other.
4. Multiply initial estimate in step 1 with EAF.

15 Different attributes called cost driver attributes determine the multiplying factors which depend on product, computer, personnel and project attributes are shown in the table below:

Cost Drivers		Rating				
		Very Low	Low	Normal	High	Very High
1	Product Attributes: RELY = Required Reliability	0.75	0.88	1.00	1.11	1.40
2	DATA = Database size	0.94	1.00	1.08	1.16	-
3	CPLX = Product Complexity	0.70	0.85	1.00	1.15	1.30
4	Computer Attributes: TIME = Execution Time Constraint	-	-	1.00	1.11	1.30
5	STOR = Main Storage Constraint	-	-	1.00	1.06	1.21
6	VIRT = Virtual Machine Volatility	-	0.87	1.00	1.15	1.30
7	TURN = Computer Turnaround Time	-	0.87	1.00	1.07	1.15
8	Personnel Attributes: ACAP: Analyst Capability	1.46	1.19	1.00	0.86	0.71
9	AEXP: Applications Experience	1.29	1.13	1.00	0.91	0.82
10	PCAP: Programmer Capability	1.42	1.17	1.00	0.86	0.70
11	VEXP: Virtual Machine Experience	1.21	1.10	1.00	0.90	-
12	LEXP: Language Experience	1.14	1.07	1.00	0.95	-
13	Project Attributes: MODP: Modern Programming Practices	1.24	1.10	1.00	0.91	0.82
14	TOOL: Software Tools	1.24	1.10	1.00	0.91	0.83
15	SCHEDE: Development Schedule	1.23	1.08	1.00	1.04	1.10

Determination of multiplying factor for different Attribute

Advance Model: In this model, effort is calculated as a function of programme size and a number of cost drivers for each phase. This model covers the limitations of both basic and the intermediate model. Both these models consider a software product as a single homogeneous entity. But most large systems are made up of several smaller sub systems. These sub systems have widely different attributes. Some may be organic type, some semidetached type of some embedded. Complexity of subsystems may be different.

There are four phases in the advance COCOMO model. They are (1) Requirements Planning and Product Design (RPD) (2) detailed design (DD), (3) Code and Units Test (CUT) (4) Integration and Test (IT)

Each cost driver is rated as very low, low, nominal, high and very high. Multiplying factors are assigned for Analyst capability cost driver (ACAP) for each phase of advanced model are tabulated below:

Rating	RPD	DD	CUT	IT
Very Low	1.80	1.35	1.35	1.50
Low	0.85	0.85	0.85	1.20
Nominal	1.00	1.00	1.00	1.00
High	0.75	0.90	0.90	0.85
Very High	0.55	0.75	0.75	0.70

Rating of Cost drivers

3.8 THE SOFTWARE EQUATION

The software equation is a dynamic multi variable model that assumes distribution of efforts over the useful life of the project. The software equation can be derived from data obtained by studying several

existing projects and it been done by collecting over 400 contemporary software projects. For the calculation of effort, the following equation has been framed:

$$E = \left[\underset{\substack{\uparrow \\ \text{LOC}}}{\text{Size}} \times B^{0.333} / P \right]^3 \times (1/t^4) \dots \dots \text{equation}$$

Where

P = Productivity parameter which indicates the maturing of overall process and management practices. This parameter also indicates:

- (a) The level of programming languages used.
- (b) The state of software environment.
- (c) The skill and experience of the software team.
- (d) Complexity of software application.

T = it indicates project duration in months or years

E = Efforts in person-months or person-years.

B = Special skills factor — the value of B increases over a period of time depending upon the importance and need for integration, testing, quality assurance, documentation and management.

Software equation has to important independent indicators they are:

- I. An estimate of size (in LOC)
- II. Time duration in months or years.

Potnam and myers suggested another set of equation derived from the software equation for simplification of estimation process.

$t_{\min} = 8.14 [LOC/P]^{0.43}$ in months for $t_{\min} > 6$ months equation-II (a)

$E = 180 B t^3$ in person – months for $E \geq 20$ person months eqn-II (b)

____For small programs with sizes between 5KD LOC and 15KD LOC, the value of B is 0.16 and for programme with sizes greater than 70KD LOC; the value of B is 0.39

3.9 PROJECT SCHEDULEING

Definition

Project Scheduling can be defined as the process of effectively managing the tasks of the project by identifying and organizing the tasks into a series of events. Project scheduling prepares details such as (i) start dated of the project, (ii) end date of the project, (iii) milestones, (iv) specifying the resources required and (v) tasks for the project. In ensures a harmonious completion of the project on time. It also helps to avoid the extra cost incurred when the project gets delayed. Project scheduling tries to predict the future. Although it is not possible to say with certainly about the length of time yet there are methods which can increase likelihood of being close.

Principles of Project Scheduling

The most commonly adopted principles for project scheduling are as follows:

- (i) **Compartmentalization:** The project is divided into several tasks to manage the project effectively.
- (ii) **Interdependency:** All the activities of the project are inter-related and not independent of each other. Various activities are performed in a sequential manner. Some activities depend so much on other activities that they cannot begin until the activity on which they depend is completed.
- (iii) **Time allocation:** Another principle of scheduling is time allocation. It determines the time for performing specified activities. It is also very important to estimate the effort required. The project management should assign final time to each team member.

- (iv) **Effort Validation:** The project manager assigns tasks to the team members according to their capabilities and efforts required for the task.
- (v) **Defined Duties and responsibilities:** The roles and responsibilities of each project team member should be decided in advance depending on their skills and capabilities.
- (vi) **Defined Outcomes:** According to this principle, the outcomes of every task should be specified. A product is outcome of a task and these products are well combined in deliverables.
- (vii) **Defined Milestones:** Mile stones be specified when task products are complete and it should be renewed for quality.

Delaying Factors

There are various factors which are responsible for delay in the project schedule. The most common factors are:

- **Unrealistic Deadlines:** Sometimes the allocation of time for completing a project is not practical because inexperienced individuals establish a wrong deadline. The project gets delayed if deadline is not achieved.
- **Change in User Requirements:** There is delay if user requirements are changed after the project has already started. This disturbs the project schedule and thus more time is taken.
- **Under Estimation of Resources:** If there is under-estimation of resources, it leads to delay in the completion of tasks of the projects.
- **Lack of Analyzing Risk Factors:** If risk factors are ignored during project planning and scheduling; these risk factors put on a bad effect on the software development.
- **No Communication among Team Members:** Lack of communication among team members makes it difficult for the project management to complete the project on time. Personal problems of team members also stand in the way of the completion of the project.
- **Lack of Action on the part of:** If the management fails to take timely action and does not understand that the project takes more time to complete; then it also becomes the reason of delay.

Activities in Project Scheduling

Various activities are involved in project scheduling. There are to be observed sincerely. The experience in the project area and in the scheduling. Those persons are interviewed who had experience with similar projects. Transition between activities is taken into account uncertain resources of talent, equipment or data are also studied so involved in project scheduling is as follow:

1. Gant Technique
2. Mile stone Scheduling
3. Pert and CPM

The Gant technique is used where production operations are highly repetitive and work performance of different departments can be combined.

In the mile stone scheduling system the milestones are developed and established in the planning phase.

PERT/Networking and CPM method are designed to be used in the development phases for identifying the critical path, float and slack.

PERT Chart (Program Evaluation and Review Technique Chart)

A PERT chart is a graphical chart meant to determine the activities that from the “Critical Path” which if delayed will harm the overall project and the project will be delayed. The use of PERT chart is justifies in large projects in which activities can be divided into two categories (i) Critical activities, (ii) Non-critical activities. The critical activities should not be delayed.

A PERT chart is a method of project management which is used to schedule, organize, and coordinate different tasks within a project. It analyses the involved tasks and determines the time needed to complete each task and thereby identifies the minimum time needed to complete the total project in time.

The PERT chart methodology was first used by U.S. Navy in 1950s to manage the Polaris Missile Project.

Advantages of PERT Chart

1. It illustrates the project in a graphic form.
2. It guesses expected time required to complete a task.
3. It provides information regarding the time expected to complete the project.
4. It describes if there is the probability of completion of the project before the date specified for it.
5. It specifies the activities forming the critical path.
6. It provides information about start and end dates of different activities.
7. It describes the inter-dependence of one or more tasks on each other.

Steps for Creating a PERT Chart

1. Identification of different activities and milestones.
2. Recognizing sequence of Activities: Different activities are inter dependent and inter-related. It is decided whether the activities are serial wise or concurrent and then relationship among activities is depicts.
3. Preparation of PERT chart.
4. Estimation of Time: An activity can be shorter or longer depending upon the time required completing it. The time estimates can be months, weeks or days hence three categories arise.

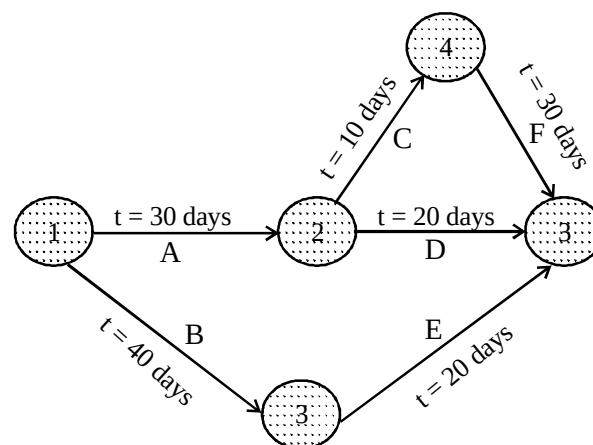
Optimistic Time: It is the shortest time needed by an activity to be completed.

Most Likely Time: It has the highest probability for the completion time required.

Pessimistic time: It is the longest time taken by an activity.

5. Critical Path Determination: Critical path for each activity is specified which determines the time required as per the project schedule.
6. Updating of PERT Chart: If any need arises; PERT chart is modified to avoid delay or when additional resources are to be provided.

Illusion of PERT Chart



PERT Chart

Fig. 3.2: PERT Chart

Milestones

In figure 1, 2, 3, 4, 5 represent the mile stones (represented by circles). The can also be represented by rectangles. When a mile stone is completed, It gets one number higher than the previous milestone. Each milestone is linked to other milestone by one or more arrows.

Activities

English letters A, B, C, D, E, F represent different activities. The direction of arrows shows the sequence of activities. There are two types of activities i.e. Serial activities and concurrent or parallel activities.

- (a) (i) Activities A, C,F are serial activities taken in the same sequence. Similarly Activities B and E are also in the same sequence.
- (ii) Activities A and B are concurrent activities. Similarly C and D activities are concurrent activities as these are performed simultaneously.
- (b) Time for each activity is depicted by 'L' in days. It can be depicted in weeks or months also.

Gantt Chart

There are various techniques of project scheduling, task method and tracking the schedule which keep an eye on the activities that are completed as per the project schedule. These techniques provide information about activities in a graphical form – through bar charts or histograms etc. It becomes easy for the management activity. The commonly followed techniques include Gantt Chart, PERT Chart etc.

Gantt chart is in the form of a graphic diagram. This chart prepares a picture of the activities of the project. The horizontal bars in the diagram show the total time span required. This time span is divided into months, weeks and days. The time taken to complete an activity is shown in parts. The vertical column shows the activities involved. The graphical chart also shows the start and end dates of each activity. This is the reason that Gantt chart is also known as timeline chart.

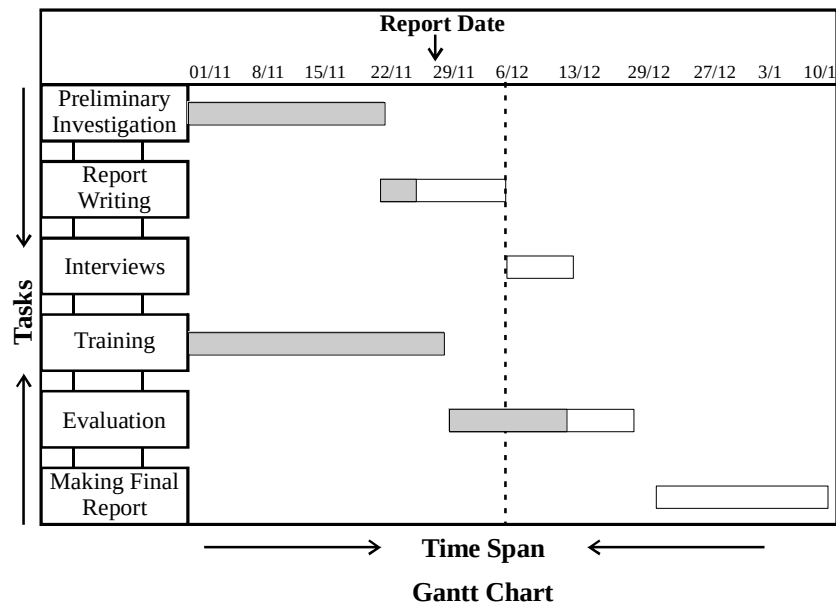


Fig. 3.3: Gantt Chart

A Gantt chart is commonly prepared on a graph paper. If the Gantt chart is big and complex then other applications like Microsoft fixed etc. A project manager comes to know about the status of the activities. Gantt chart has many advantages as given below:

1. Representation of the project in a graphical form

2. It shows the progress of each activity.
3. It provides a record of the different tasks facilities.
4. It depicts milestones after each activity is completed.
5. It clarifies the tasks which are assigned to project management team members.

There can be a change in the schedule of two projects. Hence Gantt chart also varies accordingly. The diagrammatic representation given above shows the schedule if a project:

1. The horizontal bars show the total time span in the form of months, weeks and days.
2. The time for each activity is shown in increments.
3. The vertical axis shows different tasks like preliminary members, Training and so on.
4. Shaded parts show the part of the activities that has been completed.
5. Unshaded parts show uncompleted part of the activity.
6. Horizontal bars have different lengths showing the time required to complete a specific activity.
7. Time span of one or more activities can overlap each other.
8. With the progress in the project the unshaded bars are shaded showing the completion of activities.
9. The vertical line represents the report date on which an arrow is drawn.

3.10 BASICS OF SOFTWARE MEASUREMENTS

Measurements in everyday life has a significant role to play. Without accurate measurements; we shall not be able to take right and timely decisions in area of production and manufacturing. Software engineering uses the concept of measurement in a big way to estimate costs, to monitor inventories and asses quality. This is desirable to understand and improve the software process by predicting, planning and controlling the software projects measurements are in no way an conditional or non-value added task but it has evolved into a very significant discipline in software engineering. This is the reason that most of the organizations are using measurements to ensure better quality products. Software engineers use measurements to gain insight into the design and development of the work products.

Measure, Measurements, Metrics and Indicators are the terms which are used in measuring various aspects of software. Although they have resemblances but they are different.

Measure: It means to mention some aspect of the object in a quantitative way. It can be defined as the quantitative indication of amount, dimension, capacity or size of the products and process and their attributes. It is established. When a number of errors are detected in a single review.

Measurement: The method of determining a measure a known as measurement. It is the process of assigning number or symbols are assigned to different attributes. It is established when different components are reviewed and each unit is tested to collect the measure of a number of errors in all these components. Clearly defined rules are followed.

Metrics: A metric is used to compare two or more measures. It is a quantitative measure of the particular characteristic of a program's efficiency. It also assesses the level of complexity, strength of the module and estimates time and cost of the project.

Indicator: This term is used to denote a representation of metric or a number of series of metrics that provides information about the software development project in process. The indicators enable the project manager to adjust the process, the project or the product for better results. It draws a person's attention to a particular problem, tracks potential risks and evaluates the project team's ability to control the quality of the products.

3.11 TYPES OF SOFTWARE MEASUREMENT

Some sort of measures are required to assess the quality of the final product in a software design and to better understand the models produced. The main aim is to improve the software process on a regular basis. This helps in estimation, quality control project control throughout the life cycle of a software

Project Decision making and further planning becomes easy and fruitful. There are two types of measurements:

1. Direct Measurement
2. Indirect Measurement

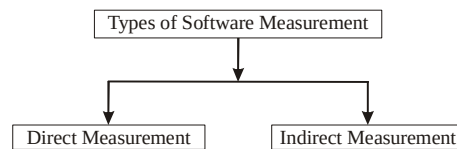


Fig. 3.4: Types of Software Measurement

3. **Direct Measurement:** When the attribute of an entity is measured directly without taking consideration of other attributes, we name it as direct measurement. Direct measures include cost and effort applied and products like lines of code, execution speed, memory size and other defects that have been found.

4. **Indirect Measurement:** There are some attributes like functionality, quality, complexity, reliability, maintainability efficiency can't be measured directly. They are calculated on the basis of the study of other attributes.

The measurements once taken help the project manager and the entire software team to take right decisions that will lead to successful completion of the project.

3.12 CRITERIA OF MEASUREMENT

The measurement program is considered successful if it follows the following key characteristics. diagram

1. **Objective and Repeatable:** The measurement program should be very easy to calculate and understand the same measurement tool should be possible to repeat again and again on similar set of data, without any problem.
2. **Timely:** The measurement program must be should be made available at the correct time to put on influence on the quality development and maintenance of the software project.
3. **Iterative:** The measurement process should be iterative which means that the project manager and team members should continually focus measurement efforts on the most critical areas. It should continually work unstopped during the project life cycle to provide necessary information and knowledge to the project managers.
4. **Related to information needs:** The whole process which includes collection, analysis and reporting of data must be related directly to the information needs of the decision taking persons. The information needs should be filled on a priority basis and addressed accordingly.
5. **Interpretation:** Measurement should enable the project manager and team members to gain insight into the design and development of the work products.
6. **FeedBack:** There must be some follow up and measurement communicator's recommendations derived from product metrics to the software team. This develops appropriate metrics for software under consideration.

3.13 PROCESS OF MEASUREMENT

A software measurement process is a management tool which if conducted effectively helps the project manager and the entire software team to take steps that lead to successful accomplishment of the project. This process is all about defining what information is needed by the decision makers. Five sets of activities are done:

1. **Formulation:** To measure and develop appropriate metrics.
2. **Collection:** To collect data to derive the formulated metrics.
3. **Analysis:** This calculates and analysis metrics and the mathematical tools are used.
4. **Interpretation:** To attain insight into the quality of the product.

5. **Feedback:** The suggestions / recommendations are forwarded to the software team.

The important stages in this process are:

(i) **Plan Measurement:** In this stage, information needs are identified and the most appropriate measure is chosen to address those needs. Only important measurements are collected which are relevant to the information needs; unnecessary ones are ignored. This stage also determines the resources and the technologies required to implement a software measurement program. An organization is interested in assigning defects, errors, cost and schedules linked to the development process.

(ii) **Perform Measurement:** In this step procedure for defined measure is documented to collect each measure. In other words we can say that this stage revolves around the collection and processing of measurement data. The procedures should be clear and repeatable. A mechanism is needed to verify the correctness and effectiveness of measurement.

(iii) **Evaluate Measurement:** This stage is concerned with the study analysis and evaluation of the main events of the process to ensure that the measurement approach and alerts about extra cost and warns about schedule over runs.

(iv) **Provide feed Back:** Preparation of Reports, Review exercises and project meetings etc. Provide valuable feedback to implement efficient measurement program. This activity ensures that measurement is amply supported at the project level and organization level.

The decision maker and concerned team members operate within these stages and in the end use the measurement information to develop the software as per requirement specifications.

3.14 BASIC CONCEPTS OF SOFTWARE METRICS

Metrics is defined as the quantitative measures that allow software engineers to identify the effectiveness and improve the quality of software process, project and product. IEEE defines a metric as “a quantitative measure of the degree to which a system, component or process possesses a given attribute”. Software metrics serve the main purpose of identifying and controlling essential parameters that have an effect on the software development. It is also helpful in fulfilling the following objectives:

- (i) Quantitative measurement of the size of the software.
- (ii) Assessment of the level of complexity.
- (iii) Assessment of the strength of the module.
- (iv) Assessment of the testing techniques.
- (v) It specifies when to stop testing and determines the date of release of the software.
- (vi) It estimates cost of resources and project schedule.

Software metrics are needed to quantify the development process and maintenance of software. We get information about the status of an attribute and help evaluate it in an objective way. It becomes easy to make modification plans. To achieve the desired goal, software metrics are used in different projects for a long period of time to obtain indicators. Software metrics help in project planning and project management activity and thus project managers and project team members become capable of taking the right path to complete the project.

3.15 IMPORTANCE OF SOFTWARE METRICS

Software Metrics have achieved a very crucial role in finding answers to questions within the discipline of software engineering. Software metrics are necessary to decide the following things :

- (i) Total time required to complete the project.
- (ii) Total cost required to complete the project.
- (iii) Number of persons and other resources required.
- (iv) Approximate maintenance cost required.
- (v) What is needed to test better quality and find the ways for better quality.
- (vi) Number of errors to be discovered before delivering the product.
- (vii) Estimation of effort to make modifications.

Software metrics help in minimising the software problems and thereby making it possible to efficiently measure and evaluate the attributes of the software. Thus we see that software metrics are helpful in giving the right information about the software development process and in this way help in determining the areas of improvement and build a successful system.

3.16 CATEGORIES/TYPES OF SOFTWARE METRICS

An effective software metrics helps a software engineer to develop software as per user requirements within estimated schedule and cost estimate etc. For this different type of metrics for measurement are there. They have been categorized on follows:

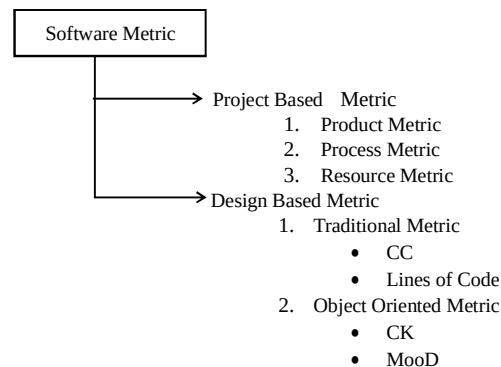


Fig. 3.5: Software Metric

3.16.1 Projects Based Metrics

Project metrics illustrate the project characteristics and their execution. Project metrics help the project manager in assessing the current projects. He can identify potential risks and problem areas. After that he adjusts work flow and evaluates the project team's ability to control the quality of work products. Project metrics are used for tactical purposes and not for strategic purposes.

Project metrics help to decrease the potential risks and problems. By making necessary adjustments; project metrics help to minimize the development schedule. This avoids delay, the other purpose of project metrics is to assess the product quality and to modify the technical issues as and when required. By reducing the number of errors and defects, the quality of the project improves and this leads to decrease in the overall cost of a software project.

In project metrics; often the first application takes place in estimation process. The study and the metrics of the previous projects help in estimating time and effort for the current project. As the project moves ahead, original estimates of time and effort are compared with the new measures of effort and time. This helps the project manager to supervise and control the project.

Project metrics are used to track the errors detected during each development phase. For example of software progresses from design to coding, project metrics are used to assess quality of the design and obtain indicators which will affect the approach chosen for coding and testing. Project metrics are used to measure rate of production in terms of models developed, function points and delivered source lines of code.

3.16.1.1 Product Metrics

At the end of each phase, a working product is developed in the software development process. Each product can be measured at any stage to check whether it is according to the required specifications or not. Metrics are developed for these products. If a product does not meet user requirements; then necessary steps are taken. Product metrics help the software engineer to detect and correct potential errors before they may result into bigger problems. Product metrics assess the internal attributes of the products so that the efficacy of the following may be judged:

- (i) Analysis, design and code model.
- (ii) Potency of test cases.

- (iii) Overall quality of the software under development.

The following different types of metrics have been devised for products in the development process:

- (i) **Metrics for Analysis Model:** These metrics study and find answers to various aspects of the analysis model like system functionality, system size etc.
- (ii) **Metrics for Design Model:** Software engineers take the help of the metrics to assess quality of design that includes architectural metrics, component level design metrics and so on.
- (iii) **Metrics for source code:** These metrics are used to assess source code complexity, maintainability and other characteristics.
- (iv) **Metrics for testing:** These help in designing efficient and effective test cases and if also evaluates the effectiveness of testing.
- (v) **Metrics for maintenance:** To assess the stability of the software product.

3.16.1.2 Process Metrics

Process metrics assess the effectiveness and quality of the software process. It also determines the maturity of the process and the amount of effort required to develop the project. Software engineers measure specified attributes to improve the quality of the project. In order to improve any process, a set of meaningful metrics are developed to measure its specified characteristics. These metrics are used to obtain indication to devise a plan for process improvement. With the help of process metrics it becomes easier for a software engineer to assess the efficacy of the software process. Three factors namely product, people and technology have a great influence on software quality and organization performance as is from the given diagram:

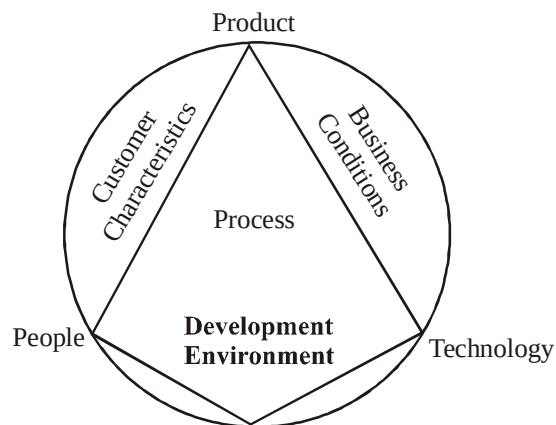


Fig. 3.6: Effect of Factors On Software quality & Organization

The skill and motivation of people, the complexity of the product and the technology level have a great influence on the quality and team performance. For measuring the desired qualities in a software process, a set of metrics is formulated based on different outcomes derived from the process. These outcomes include the following:

- (i) Number of errors detected before the release of the software.
- (ii) Defects found by the user and reported after the delivery.
- (iii) Time spent in the fixation of errors.
- (iv) Work products delivered.
- (v) Human efforts used
- (vi) Time expended
- (vii) Conformity to schedule
- (viii) Wait time
- (ix) Number of modifications
- (x) Estimated cost compared

Types of process metrics: The process metrics are of two types —

- (i) Private Metrics,
- (ii) Public Metrics.

Private Metrics : This type of metrics are private to the individual and serve as an indicator only for the specified individual(s) e.g. defect rates by a software module and defect errors by an private individual.

Public Metrics: If includes that information which is private to both individuals and teams. E.g. project level defect rates, effort and related data collected analyzed and assessed to get indicators that help in organizational process performance.

Process Metrics Etiquette: Process metrics can be beneficial because the organization works to improve its process maturity. These metrics if mis-used can create problems for the organization. The following are the guidelines to be used by managers and software engineers to avoid misuse of the metrics.

Guidelines

- (i) Rational thinking and sensitivity toward organization should be the first priority.
- (ii) Feed back to the individuals or teams is very necessary to be provided.
- (iii) Metrics should neither appraise nor threaten individuals.
- (iv) Use of single metrics should be avoided.

As the process goes on, the derivation of simple indicators lead to a stringent approach called statistical software process improvement (SSPI). SSPI uses software failure analysis to know about all errors and defects faced during the development.

3.16.1.3 Resource Metric

Resource metrics is a source code metric and quality analysis tool unlike any other on the market. It provides a standard method for analyzing C, ANSI C++, C# and Java source code across operating systems.

The unique ability of Resource Metric to support virtually any operating system provides enterprise with the ability to standardize. The measurement of source code quality and metrics throughout organization. Resource standard metrics provides the fastest, most flexible and easy to use tool to exist in the measurement of code quality and metrics.

Example Resource Metrics

- * Effort expended
 - On tasks with a project, classified by life cycle phase software function.
 - On extra project activities.
 - Training
- * Elapsed Time
- * Computer Resources

3.16.2 Design Based Metrics

The goal of having metrics for the software design is to assess the quality of the design which finally assesses the quality of the product. The quality of the intermediate product is of interest in the development process because it is believed that this leads to a high quality product. The software design complexities are too many and knowledge collected from various fields can be useful in designing systems. It is important to develop software metrics from which meaningful indicators can be derived. With the help of these indicators necessary steps are taken to design the software keeping in view user requirements. Various design metrics such as architectural design metrics, metrics for object oriented design (MOOD), Component Level Design Metrics and user interface design metrics are used to indicate the complexity, quality and so on.

Design metrics can be extracted much earlier in a project and if helps to predict greater number of activities. Large number of Recent Software Design Metrics can be divided into three categories—(i) Network Metrics, (ii) Stability Metrics, (iii) Information flow Metrics. Software metrics measures the product under development and after a system are fielded. Design Metrics includes three activities:

(i) **Total Number of Modules:** This metric specifies the total number of modules. The fundamental benefit of this metric is that by using an average size of a module, a size estimate of the final product can be ascertained. If there is difference in the initial size estimate, it will require a change in plan especially allocation of personal, the schedule and the cost of the product etc. and it will have to be negotiated with the client.

(ii) **Number of modules called:** This metric is computed easily after the completion of the software design. For each module, we can calculate how many other modules it calls and this can also be used to determine how many modules call a particular module. A module with high fan in may point that module has functional cohesion of module with high fan out means that the module depends on to many modules where as a high fan in means that a number of other modules depend on this. During software design, the designer must evaluate those modules that are called by a large number of module, or that call a large number of modules.

(iii) **Number of Parameters:** The software a design specifies all those modules that will be present in the system. It also specifies then inter-connections with other modules. When the software design reaches its completion; a number of parameters of a module can be obtained. Some questions arise about the constitution of a parameter. Whether a complex object in a parameter list be counted as a single parameter? or it should be considered to be made up of many parameters ?

The simplest way is that to consider each logical entity in the parameter list as a parameter. The second way is to think about each basic data which is used in a module as a parameter.

This metric in a way tries to find “Coupling between Modules”. When the module performs so many functions and the parameter list is very large; then a high number of parameters can have low cohesion. Modules with higher number of parameters will be more tightly coupled with their caller modules than those with a small number of parameters.

This means that more effort and time will be needed to understand such modules. Any modification in them will affect other modules as well so the job of the maintainer becomes a bit difficult. It is clear that module with large number of parameters needs careful scrutiny.

3.16.2.1 Traditional Metrics

Traditional metrics are applied to the methods that carry out the operations of a class in an object oriented system. The method is a component of an object that operates on data after receiving the message and is defined as part of the declaration of a class. This method demonstrates how a problem is broken into segments and the capabilities other classes expect of a given class. The following three metrics given below are traditional metrics — (i) Cyclomatic Complexity, (ii) Size Metric, (iii) Comment Percentage

The traditional metrics have been widely used. Researchers and practitioners well understand them. The traditional metrics are related to software quality attributes which have been validated.

Metric–1: Cyclomatic Complexity: It is used to evaluate the complexity of an algorithm in a method. A method with low Cyclomatic complexity is considered better. Due to inheritance, cc cannot be used to measure the complexity of a class. Cyclomatic complexity of individual methods can be combined with other measures for the evaluation of complexity of the class. Although this metric is used to evaluate quality attribute complexity; it is also related to all of the other attributes. The formula for testing is equal to the number of edges minus the number of nodes plus 2. Number of independent test paths = Edges – nodes + 2. Number of independent test paths = Edges – nodes + 2

Metric–2: Size Metric: Size of a class is used to evaluate the ease of understanding of code by developers and maintainers. Methods to measure size can be counting all physical lines of code, the number of statements and the number of blank lines. Coding language and the complexity of the method form the basis for evaluating the size measures. Size affects the ease of understanding, routines of large

size will always pose a higher risk in the attributes of understanding, Reusability and Maintainability. The size metric include — Total lines of code, Total function calls, Number of windows.

Metric—3: Comment Percentage: The line counts which compute the size metric can be expanded further to take into account a count of the number of comments both on line i.e. with code and stand alone. The comment percentage is equal to the total number of comments divided by total lines of code less the number of blank lines. This metrics is used by Developers and Maintainers to evaluate the attributes of Understandability, Reusability and Maintainability.

3.16.2.1.1 McCabeCyclomatic Complexity (CC)

- It is a software metric developed by Thomas J. McCabe in 1976. It is a very useful, logical metric. It tests the complexity of the program. It also estimates the amount of effort required to understand the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable.

- It measures the number of linearly independent paths through a source code. Cyclomatic complexity is calculated using the control flow graph of the program. The nodes of the graph correspond to indivisible groups of commands where as a directed edge links two nodes, if the second command immediately follows the first command.

- Cyclomatic complexity may also be applicable to individual functions. CC is an indication of the number of 'Linear' segments in a method. A method with no branches has a cyclomatic complexity of 1 since there is 1 arc. This number is incremented whenever a branch is encountered. CC is procedural rather than an OO metric.

- Cyclomatic complexity helps in estimation of complexity of code, testing effort and program reliability.

- **The statements which represent branching as:** 'For', 'while', 'do', 'if', 'case', 'catch' optional and ternary operator (optional). If the source code contained no decision points like 'IF' statements or 'FOR' loops, the complexity will come out to 1 because there is only one path. But if the code had a single IF statement, there would be two paths through the code:

- (i) First path where the IF statement is evaluated as TRUE.
- (ii) Second path where the IF statement is evaluated as FALSE.

- **The formula of the cyclomatic complexity is defines as:**

$$CC = E - N + P$$

CC= Cyclomatic Complexity

E= Number of Edges

N= Number of Nodes

P= Number of Connected Components

Example 1:

If number of edges= 7, number of nodes= 6 and number of connected components= 2.

*Then cyclomatic complexity of **figure 5.4** is:*

$$CC = 7 - 6 + 2 \Rightarrow CC = 3$$

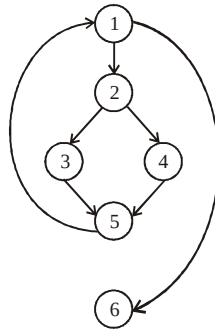


Fig. 3.7: Example of Cyclomatic Complexity

Example 2:

If number of edges= 10, number of nodes= 8 and number of connected components= 1
 Then cyclomatic complexity of **figure 5.5** is:
 $CC = 10 - 8 + 1 \Rightarrow CC = 3$

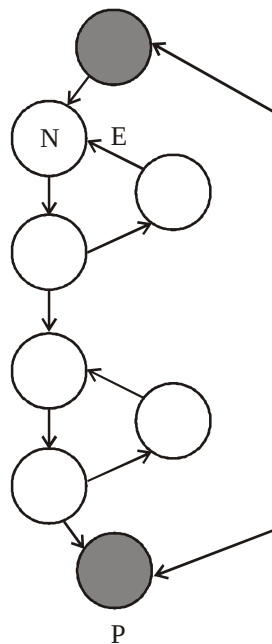


Fig. 3.8: Diagrammatic representation Structured Program

- It is clear CC is calculated by creating a connected graph of the source code. Each line of code is considered as a Node and the arrows between the nodes show the Execution Path. After cyclomatic complexity of a code has been calculated, then we can compare its complexity of other programs as per the standard range tabulated as under:

Cyclomatic Complexity

- (a) 1-10
- (b) 11-20
- (c) 21-50
- (d) $CC > 50$

Code Complexity

- It is a simple program with less risk
- More complex program with moderate risk
- Complex with high risk
- Untestable having very high risk

- Second method to determine the metric is to count the number of decision points (conditionals) and adding

$CC = D + 1$ Where D is the number of decision points in the source code.

Advantage of McCabe Cyclomatic complexity

- (i) It can be used as a case of maintenance metric.
- (ii) When it is used as a Quality Metric, gives relative complexity of various designs.
- (iii) It can be computed early in life cycle of Halstead's metrics.
- (iv) It measures the minimum effort and best areas of concentration for testing.
- (v) It guides the testing process by limiting the program logic during development.
- (vi) It is easy to apply.

Disadvantages

- (i) It does not measure the Data Complexity. It measures only the program's control complexity.
- (ii) The same weight is placed on nested and non-nested loops. However, deeply nested conditional structures are harder to understand than non-nested structures.
- (iii) CC may give a misleading figure in the case of simple comparisons and decision structures.

3.16.2.1.2 Source Lines of Code

SLOC is the most common metric or size today and is most widely used for size estimation. Line of code is easy to compute and enjoys some reputation. It highly depends on programming language because code writing varies from one programming language to another. Simple line metrics like errors per LOC, defects per LOC, cost per LOC etc. can be derived from LOC. Lines of code has also been used to pre-guess program complexity, development effort, and programmer performance.

LOC is the earliest and simplest metric for estimating the effort and size of a computer program even then there is no standard definition of LOC. It is due to this reason, different workers may obtain different counts rural emphasis is given to each line of code. LOC is often used during the testing as well as maintenance phases.

Advantages

It is very simple to measure.

Disadvantages

- (i) It is dependent on programming language.
- (ii) It does not accommodate non-procedural language.
- (iii) Poor software design may lead to excessive and unnecessary line of code.

Guidelines for determining LOC

- (i) One LOC is for one logical line of code.
- (ii) Only those line of code are included which are delivered as part of software.
- (iii) Test drivers, test stubs and other support software are excluded.
- (iv) The software code written by the software developer is included whereas code created by the application generator is excluded.
- (v) All declarations in the programs are counted as lines of code.
- (vi) No comment is counted as lines of code.

3.16.2.2 Object Oriented Metrics

Source line of code and Functional Point Metrics can be helpful in estimating object oriented software projects. But these metrics do not succeed in providing details for effort and schedule estimation in incremental software development. Different types of metrics have been suggested for object oriented projects which have been given below:

- (i) **Number of Scenario Scripts:** These are a series of steps which depict the interaction between user and application. There is direct relationship between number of scenarios and the application size

and number of test cases. Which are developed to test the software. The scenario scripts are analogous to use cases.

(ii) **Number of key classes:** Key classes are independent components. The key classes give indications about effort required to develop software and the amount of 'reuse' feature to be used during the development process.

(iii) **Number of support classes:** Support classes can be defined as those classes which are needed to implement the system but are related indirectly to the problem domain. E.g. user interface classes, computation class. For each key class, a support class can be developed. Support classes and key classes function in a similar way.

(iv) **Average number of support classes per key class:** Key classes are defined very early in the software project whereas support classes are defined throughout the project. If average number of support classes per key class is pre-determined; then estimation process gets simplified.

(v) **Number of sub systems:** Sub system can be defined as a collection of classes that is supporting a function visible to the user. After the sub systems are identified it becomes easy to prepare on appropriate schedule, according to which work on sub systems is divided among project members.

Above mentioned projects along with other metrics like effort used, errors and defects detected are collectively used.

3.16.2.2.1 CK (Chidamber&Kemerer's Metrics Suite)

CK metrics for OO Design is the deepest research / study in object oriented investigation. The following six metrics have been defined for it.

Metric 1 :Weighted Methods per class (WMC) : Let us suppose a class C1

Where M1.....Mn are the methods.

and C1.....Cn be the complexity of the methods

Then the formula for WMC = $\sum_{i=1}^n C_i$

The following points are to be noted:

- * The number and complexity of the method, give an indication of Time and Effort required developing and maintaining the class.
- * The number of methods in a class is proportional to the potential impact on children. Larger the number of methods, greater is the potential impact.
- * Classes having larger number of methods are likely to be more application specific thereby limiting the possibility of reuse.

Metric 2 : Depth of Inheritance (DIT) : In those cases where multiple inheritance is involved; DIT is the maximum length from the node to the root of the tree :

- * If a class is deeper in the hierarchy, it inherits greater number of methods and makes it more complex to predict its behavior.
- * Deeper trees constitute greater design complexity.
- * The deeper a particular class in its hierarchy; the more is potential reuse of inherited methods.

Metric 3 : Number of Children (NOC) : It is defined as the number of immediate sub classes which are under a class in the class hierarchy and are going to inherit the methods of the parent class :

- * If NOC is greater; the Reuse will also be greater because inheritance is a form of Reuse.
- * If NOC is greater; there are greater chances of improper abstraction of the parent class.
- * NOC is linked to potential influence of a class on the design. Greater NOC in a class may require.

Metric 4 : Coupling between object classes (CBO) : Classes having same properties are said to be :

- * Excessive coupling between object classes has a bad effect on the Modular Design and prevents Reuse.

- * Inter object class couples should be minimum to improve Modularity and Promote encapsulation. As the number of couples becomes large; the sensitivity to any change in other parts of the design also becomes large so the maintenance becomes more difficult.
- * The higher the inter object class coupling, the more rigorous the testing needs to be.

Metric 5 : Response for a class (RFC) :

- * It indicates number of methods that can be invoked in response to a message.
- * RFC measures potential communication between classes.
- * When RFC Grows; the complexity of a class becomes more and more whereas understandability decreases.
- * In case of a large number of methods invoked in response to a message; the testing and debugging of the class becomes more complicated and it becomes more difficult for a tester to understand.

Metric 6 : Lack of cohesion in methods (LCOM) : LCOM can be defined as the count of the number of methods pairs whose similarity is 0 minus the count of methods pairs whose similarity is not zero. As the number of similar methods in a class becomes more and more; the cohesiveness increases:

- * Cohesiveness promotes encapsulation.
- * Lack of cohesion means that classes should be split into two or more than two sub classes.
- * Low cohesion increases complexity and it increases the chances of errors during the development process.
- * If there is any measure of disparateness of methods; it helps to identify flaws in the design of classes.

3.16.2.2.2 Mood (Metrics for Object Oriented Design)

The metrics for Object Orient Design refers to the basic structural mechanism of the OO paradigm as encapsulation (MHF, DHF) inheritance (MIF and AIF), polymorphism (PF), message passing (CF) and are expressed as Quotients. It includes the following metrics:

1. **Method Hiding Factor (MHF) :** It is defined as the ratio of sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration. The invisibility of a method is the %age of the total classes from which this method is not visible.
2. **Attribute Hiding Factor (AHF) :** It is the ratio of sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration.
3. **Method inheritance factor (MIF) :** MIF is defined as the ratio of the sum of the inherited methods in all classes to the total number of available methods (locally defined plus inherited) for all classes of the system under consideration.
4. **Attribute inheritance factor (AIF) :** It is defined as the Ratio of the sum of inherited attributes to the total number of available attributes (locally defined + inherited) for all classes of the system under consideration.
5. **Polymorphism Factor (PF) :** It is the ratio of the actual number of possible different polymorphic situation for all classes c_i & the maximum number of possible distinct polymorphic situations for class c_i .
6. **Coupling Factor (CF) :** It is defined as the ratio of the maximum possible number of couplings in the system to the actual number of couplings not computable to inheritance.

3.17 SOFTWARE METRIC APPROACH

Software Metrics are numerical data linked to the development of the software. They have the following four functions to play:

- (i) **Planning:** Software metrics lay the foundation for cost estimation, resource planning, scheduling, and budgeting.
- (ii) **Organizing:** A project's organization is influenced by size and schedule metrics.

- (iii) **Controlling:** They provide an important tool to supervise, monitor and control the progress of various development activities.
- (iv) **Improving:** Software metrics are used to ensure a quality product in the end. This is achieved by continuous process improvement.

The following steps are required to select, design and implement software metrics so that a light software metric program is developed:

- (i) **Identify Metrics Customer:** This is the first step. Those people are identified who will be using metrics for decision making and implement actions. These customers can be programmers, Testers, Experts and users.
- (ii) **Select Measurable Goals:** The second step is to select goals at the organization level and the project level. At the organization level these may be low cost provider, higher level of customer satisfaction and profit margins etc. At the project level the goals can be to deliver the software product in time, completion of project within budget and to achieve the desired level of quality.
- (iii) **Ask Questions:** Some questions are raised and answers to satisfy these questions are found which will ensure the achievement of the goals. The questions can be:
 - (a) Whether the software product adequately tested or not?
 - (b) Have all the errors properly corrected?
- (iv) **Select Metrics:** The further step is the selection of the metrics suitable to provide information to the people that helps them to take right decisions to ensure the achievement of goals after solving problems.
- (v) **Select a Measurement Function:** A measurement function is very necessary. This defines the way how to calculate the metric. Some metrics are measured directly by variables.
- (vi) **Collect Data :** After taking the decisions about what to measure and how to measure, the data is collected accordingly to compute the metrics.
- (vii) **Provide feedback :** After the computation of the metrics, the need arises to provide feedback by evaluating the metrics. The patterns indicators and goals identified by the metrics are used to determine the need for action a further investigation or to describe the level of confidence in a given result. Software metric approach is better depicted in the following diagram:

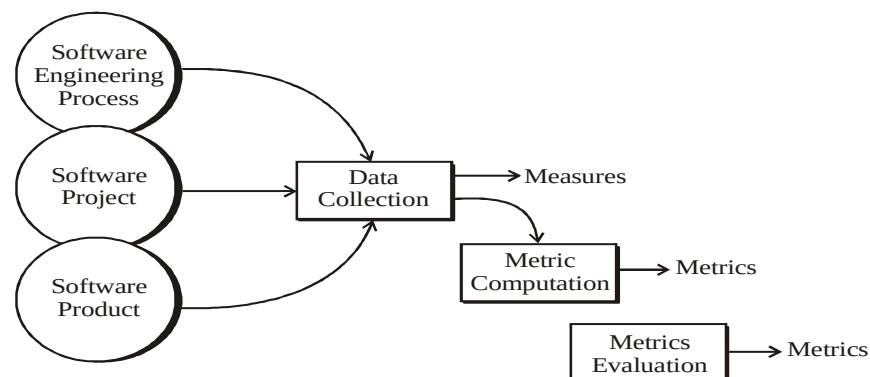


Fig. 3.9: Steps for developing Software Metric program

3.18 GUIDELINES FOR METRICS

An ideal software metrics is easy to understand, effective and efficient. An ideal software metrics is always validated and characterized effectively. The following guidelines should be followed to develop metrics:

- (i) **Simple and computable:** The derivation of software metrics should be very easy to understand and it should take average amount of time and effort.
- (ii) **Consistent and objective:** Very clean results should be produced by software metric.

- (iii) **Consistency of use of units and dimensions:** Mathematical calculations in metrics should carry the use of dimensions and units in a consistent manner.
- (iv) Programming language should be independent and the metrics should be developed on the basis of Analysis Model, design model or program's structure.
- (v) **High Quality Product :** It should lead to the production of high quality software product.
- (vi) **Easy to Calibrate :** Metrics should be easy to adjust according to the requirements of the product.
- (vii) **Easy to obtain:** Metric should be easy to be developed at a reasonable cost.
- (viii) **Robust :** Metrics should be relatively insensitive of small change in process project or product.
- (ix) **Value :** The value of metrics should be proportional to the value of software characteristic they represent. If one increases or decreases, the other should also increase or decrease relatively.
- (x) **Validation :** Metrics should be validated before it is being used for making decisions.
- (xi) **Focus :** The main focus of the metrics should be on the product and process.
- (xii) **Never to use a single metric:** Software is complex and multifaceted. One should not obsess to much on a single metric. Other metrics should not be avoided.
- (xiii) Metrics data should not be ignored or considered negative.
- (xiv) Metrics should not be used to threaten the individual or teams.

These people might manipulate the data, the next time. Metrics should never be used as a 'stick'. These guidelines when followed in true sense result into a very high quality metrics.

3.19 QUALITIES OF GOOD SOFTWARE METRICS

A Good software metrics should possess the following characteristics:

- (i) **Simplicity:** A good software metrics is very to understand. It should not be complicated.
- (ii) **Computability:** It should be easy to compute and derive the metrics. It should not require a lot of time and effort for computation.
- (iii) **Objectivity:** It should be of objective and should not be affected by subjective judgments. It should provide clear cut results.
- (iv) **Consistency:** The software metrics should show consistency so that even an independent third party should be able to derive the same metric value and using the same type of information.

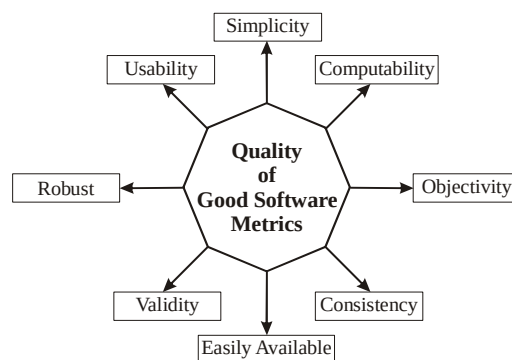


Fig. 3.10 Quality of Good Software Metrics

- (v) **Easily Available:** It should not be too much expensive. It should be available at a reasonable cost.
- (vi) **Validity :** The metric should measure what is required if the metric should provide valid result.
- (vii) **Robust :** If there are minor changes in the product; the metrics should be able to withstand. The changes which can be accidental or intentional security breaches.
- (viii) **Usability :** Software metric tries to measure or predict some attribute of some product or process. The software should be user friendly in order to avoid user's reluctance and failures.

3.20 SOFTWARE QUALITY

1. Each type of customer will have their own definition of quality. It means quality definition may differ from person to person. For example: an organization defines quality in terms of profits, whereas user defines quality as bug free, within budget and user friendly.

2. Quality can be defined as:

- **According to Oxford dictionary: Degree of excellence**
- **According to Edward Deming: Fitness for purpose**
- **According to ISO: The totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs.**

3. Quality of software depends upon many factors like bug-free, portability, reusability, delivered on time, within budget, user friendly, meet user's requirements and maintainability.

4. It is a relentless process means ongoing process.

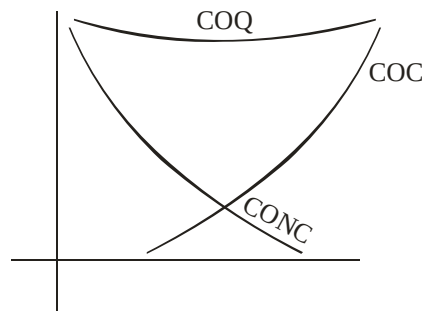
5. The quality system activities encompass the following:

- Audits of projects
- Review of System
- Development of standards and guideline.

6. Cost of quality depends upon the level of testing and maintenance.

Cost of quality = cost of conformance + cost of non-conformance

$COQ = COC + CONC$



7. Need of Software Quality

- (a) If a software has a high quality, it means software has fewer defects. This saves time during testing and maintenance phases.
- (b) Higher the quality of software, lower the maintenance cost. It increases the reliability and customer satisfaction.

3.21 ATTRIBUTES OF QUALITY SOFTWARE

The attributes are those factors which increases the quality of a software which are as follows:

1. Bug free
2. Portability
3. Reusability
4. Delivered on time
5. Within budget
6. Customer Satisfaction
7. Maintainability
8. User friendly

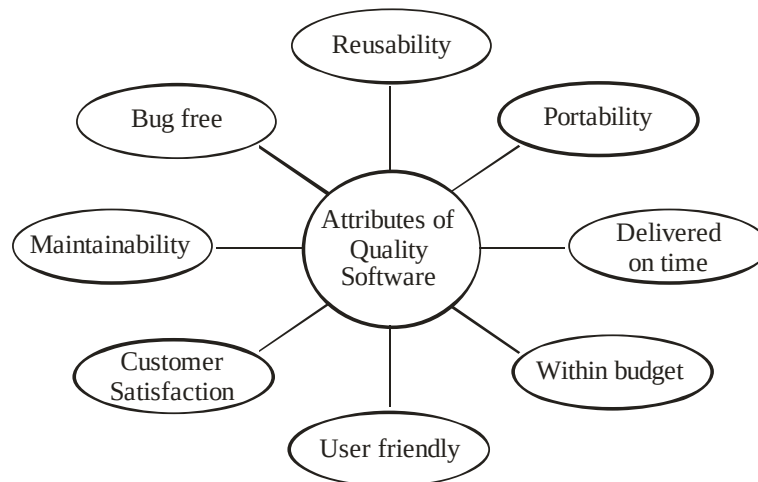


Fig. 3.11 Attributes of Quality Software

1. **Bug-free:** A quality software is always bug free. A software is a quality software if there are lesser number of bugs in the software product.
2. **Portability:** Portability is an attribute in which a software product can be adopted to run on computers (machines) other than the one for which it was designed. According to this feature, a quality software can run on different machines, different operating system environments and with other software products.
3. **Reusability:** According to this feature, we can make some changes in the software product to build a new version of the same product. In this process, minor changes are made in the old software product to make a new software product. In this process, different modules of the product are, reused to develop new product.
4. **Delivered on Time:** A quality software is always delivered on time. It means software is delivered in a fixed time period and within budget.
5. **Within Budget:** A software is called a quality software, if it is under budget of a client. If a client/customer cannot afford a software because of its high cost, then that software is not a quality software for that particular client/customer. A quality software should be within budget of a customer.
6. **User Friendly:** A software product is user friendly if its users find it easy to use. A quality software is always user friendly. A software with many features is never called a quality software if a user faces problems to use it. In a quality software product, users can easily invoke the functions and features of software product.
7. **Customer Satisfaction:** It is another important attribute of software quality product. A software is known as quality software for a particular user if it meets the user's requirements. Software should satisfy the needs/requirements of the customer/user.
8. **Maintainability:** A quality software is maintainable if errors can be easily corrected when they show up, functions of the software product can be easily added to the same software product. This attribute provides the suitability for debugging, extension of functionality and ease of modification.

3.22 SOFTWARE QUALITY MANAGEMENT SYSTEM

1. Software quality management system is also known as quality system. Software quality management system ensures that organizations use an adequate methodology to develop a desired quality software.
2. Software quality management system is the responsibility of whole organization. Software quality is not a responsibility of individual person. The top management of the organization should actively participate in this system. If top management of the organization will actively participate in the system, then every member of the organization will take the quality task seriously.

3. Software quality management system involves many activities like auditing and reviewing of the quality system, development of standards, procedures and guidelines etc to enhance the quality of software product.

4. In the software quality management system, finished products are inspected to eliminate the defective products.

5. This process has mainly four stages which is known as quality assurance method.



Fig. 3.12 Quality Assurance Method

In the first stage, the product is examined from each angle. In the second stage, quality control finds the defects and eliminates the defects. Quality control also determines the reasons behind the defects and corrects the reasons of defects. In the third stage, quality assurance takes place. The planned systematic activities necessary to ensure that a component, module or system conforms to established technical requirements. The modern quality paradigm includes certain guidance for recognizing, defining, analyzing and improving the product process. In the last fourth stage, quality management aims at continuous process improvement.

3.22. 1 Software Quality Management Principles

The principles of software quality management system are as follows:

Principle 1: Organizations should clearly understand the needs of current and future customers. The aims of the organizations should meet the customer requirements.

Principle 2: Top management of organization should create such an environment in which all people can become fully involved in achieving the organization's common goal.

Principle 3: All people of organization should actively involve in achieving the organization's goals and benefits.

Principle 4: All resources and activities of an organization are managed efficiently to achieve common goal.

Principle 5: Organizations should take effective decisions and actions to achieve desired goals.

Principle 6: Achieving quality is a relentless process means ongoing process.

3.23 SOFTWARE QUALITY ASSURANCE (SQA)

1. Software quality assurance (SQA) is a planned and systematic approach to the evaluation of the quality.
2. SQA ensures that the standards, processes and procedures are established and followed throughout the software development life cycle.
3. In the software quality assurance, processes and methods are monitored throughout the software development life cycle.
4. The goal of quality assurance is to prevent introducing defects in the software application which helps to improve the development process. Its aim is prevention of defects to improve the quality.
5. Software quality assurance is a pro-active process. It identifies the weaknesses in the process but it does not involve in the execution of program.
6. All people who are involved in the software development process are responsible for quality assurance.
7. Verification is an example of quality assurance.

3.23.1 Importance of Software Quality Assurance (SQA)

Software quality assurance (SQA) is an important process in the software development approach to enhance the quality of software product. It is a proactive process used to prevent the defects and to improve the quality of the software.

The following are some points which explains the importance of SQA:

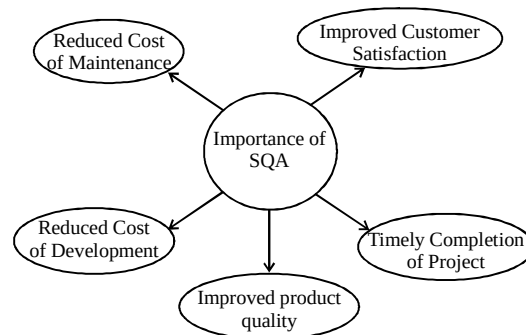


Fig. 3.13 Importance of SQA

1. **Improved Customer Satisfaction:** SQA ensures that the developed software meets the customer's requirements and fulfill their desired goals. SQA ensures that software product does what it is supposed to do. Customer satisfaction is the most important factor, for which quality assurance is done.
2. **Timely Completion of Project:** In SQA process, all testing and development processes go smoothly and quickly. SQA assures that the software product is free from errors/bugs/defects. It means that software development project consistently reach towards completion within budget and time.
3. **Improved Product Quality:** The quality of software product gets improved by following standards, procedures and actions. The aim of quality assurance is to prevent introducing defects in the software product which improve the product quality.
4. **Reduced Cost of Development:** In this process, developers are working with SQA team. Both developers and SQA team monitors the software development process from planning to implementation stage to maintain quality. In this way, there are lesser chance of an error to come up in the software development process. The developers don't need to go back to the application and fix something all the time and cost of development is automatically reduced.
5. **Reduced Cost of Maintenance:** If a software product is properly quality assured it means software is bug free. It will give less chances to users to complain and time spent on maintenance will be much less.

3.23.2 Activities of Software Quality Assurance(SQA)

The Software Quality Assurance is the process of evaluating the quality of a product. It is an umbrella activity that ensures conformance to standards and procedures throughout the SDLC of a software product. For the enhancement of quality in the software product, the following activities are performed:

1. Formulating a quality management plan
2. Applying software engineering techniques
3. Conducting formal technical reviews
4. Applying a multi-tiered testing strategy
5. Enforcing process adherence
6. Controlling change
7. Performing SQA audits
8. Keeping records and reporting

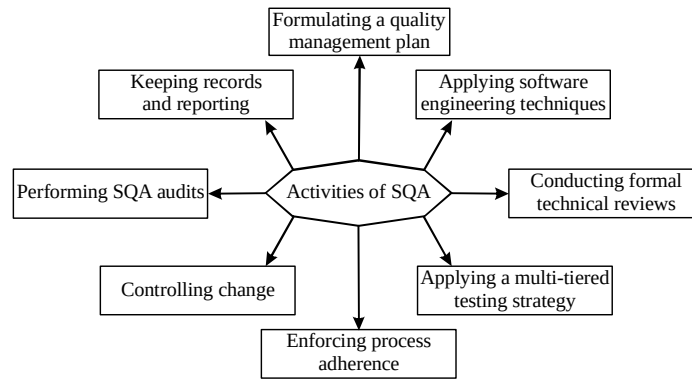


Fig. 3.14 Activities of SQA

1. **Formulating a Quality Management Plan:** The first task of SQA is the formulation of a quality management plan.
 - The quality management plan identifies the quality aspects of the software product to be developed.
 - Plan check points for work products and the development process.
 - Tracks changes made to the development process based on the results of the checks.
2. **Applying Software Engineering:** The software engineering theories and practical techniques help the software designer to achieve high quality specification.
 - Information gathering techniques help designer to collect information.
 - The project estimation techniques calculate the estimation of the project.
3. **Conducting Formal Technical Reviews (FTR):** The formal technical review (FTR) is conducted to perform following activities:
 - To assess the quality and design of the prototype.
 - It is a meeting with the technical staff to discuss the quality requirements of a software product and its design quality.
 - It helps in detecting errors at an early phase of development.
 - It prevents errors for moving down to the latter phases and resulting in rework.
4. **Applying a Multi-tiered Testing Strategy:** The software testing is a primary task of SQA activity which is described as
 - Error detection and correction in project.
 - Different types of testing is performed in sequence, first of all unit testing is performed and in the subsequent levels, integration testing and system level testing are applied.
5. **Enforcing Process Adherence:** It emphasizes the need for process adherence during product development. It is a combination of two tasks, product evaluation and process monitoring.
6. **Product Evaluation:** The product evaluation is conducted to ensure the following:
 - To ensure that the standards laid down for a project are followed or not.
 - To ensure that the compliance of the software product to the existing standards is verified.
 - To ensure that the software product reflects the requirements identified in the project management plan
7. **Process Monitoring:** The process monitoring ensures and monitors the following activities:
 - Does the appropriate steps to follow the product development procedures are carried out.
 - Monitors processes by comparing the actual process with the written documented procedures.
 - It ensures that the development and control processes described in the project management plan are correctly carried out.
 - It ensures that products and processes conform to standards and procedures.
 - Audits ensure that product evaluation and process monitoring are performed.

8. **Controlling Change:** It evolves both human and automated procedures to provide a mechanism for change control. The change control mechanism is implemented during the development and maintenance stages.

- It ensures software quality by formalizing requests for change.
- It evaluates the nature of change.
- It controls the impact of change.

9. **Performing SQA Audits:** The SQA audits scrutinize the software development process by comparing it to established processes.

- It ensures that proper control is maintained over the documents required during SDLC.
- It ensure that the status of an activity performed by the developer is reflected in the status report of the developer.

10. **Keeping Records and Reporting:** The keeping records and reporting involves the process of collection and circulation of information relevant to SQA. The results of reviews, audits, change control, testing, and other SQA activities are reported and preserved for future reference.

3.23.3 Software Quality Metrics

Software quality metrics are used to express the quality of software product. No single metric is there which appropriately explains the quality of the software product. Software quality metrics are some measurements which are related to system, process and documentation of software. Software quality metrics are divided into three types which are as follows:

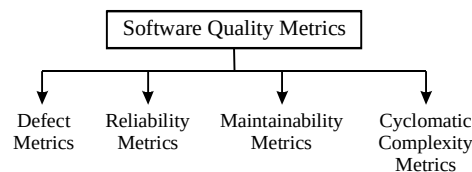


Fig. 3.15 Metrics of Software

1. **Defect Metrics:** Defect metrics provide the number of defects observed in a particular software product. It is used to reflect the man-hours spend during testing to resolve the defects. Defect metrics also explain the types of defects like computation defects, usability defects, requirement objects, design defects etc. to prevent the future problems.

2. **Reliability Metrics:** Reliability metrics are used to know the probability of software failure or the rate at which software will occur. Reliability metrics are discussed in chapter 11. 3.

Maintainability Metrics: Maintainability metrics are used to measure the maintainability of the software product. The maintainability of a program is related to its complexity. The complexity is measured by assuming no. & frequency of operators, no. & frequency of operands.

4. **Cyclomatic Complexity Metric :** It is a very useful and logical metric. It tests the complexity of the program. It also estimates the amount of effort required to understand the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable. It measures the number of linearly independent paths through a source code. Cyclomatic complexity helps in estimation of complexity of code, testing effort and program reliability.

3.23.4 Principles of SQA

The SQA team has to follow some principles which ensure that the application will live up to the expectations of the users. Some commonly known important principles which are used for proper execution of software quality assurance (SQA) are as follows:

1. **Feedback:** The time is always a best friend and biggest enemy of any developer. If SQA team give feedback early to the developer, then developer can easily take action. In short, faster the feedback, faster the development will move forward.

2. **Motivation:** Quality assurance is a very tedious task. SQA team members should be motivated, passionate and have right mindset so that they can work efficiently, effectively and with creativity.

3. **Evolution:** This principle is for future use. SQA team should be able to mark every time something new is done. Every time something new happens, it should be always noted.

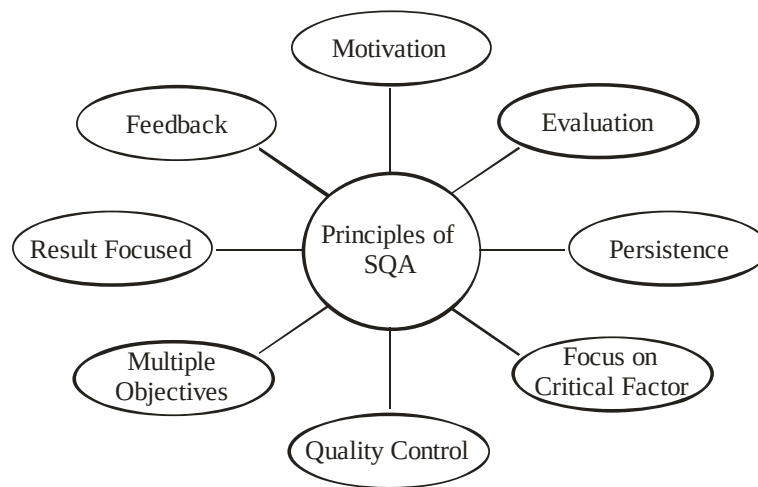


Fig. 3.16 Principles of SQA

4. **Persistence:** The SQA team members should be very patient in every aspect of the software development process. There is no perfect application and every part should be scrutinized without hesitation.

5. **Focus on Critical Factors:** There are many factors developed in the software are not as critical compared to others. SQA team members should focus on more critical areas first.

6. **Quality Control:** Quality control is the pillar of quality assurance. Everything needs to have quality control from start to the finish. The biggest and tightest quality control should be executed as early as possible which emphasizes on where to start.

7. **Multiple Objectives:** This principle is a challenge and risk for SQA team members. At the start of SQA planning, the SQA team members should have more than one objective and each objective should be focused on.

8. **Result Focused:** The SQA process should always look for results whenever a phase is set. SQA should be result focused because it affects the clients and users.

3.23.5 Software Quality Assurance Standards

The following are some standards which assure the quality in a software product:

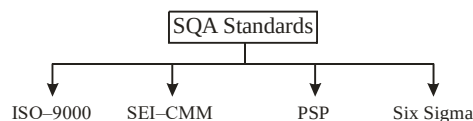


Fig. 3.17 Standards of SQA

I. ISO-9000

1. ISO is also known as International Standard Organization. It is a non-governmental organization. It is a network of 63 countries which is established to formulate standardization. In 1987, ISO published its 9000 series of standards. ISO 9000 interprets the ISO standards for software industry in 1991 because manufacturing industry and software industry have different standards. In India, ISO certification is offered by BIS (Bureau of Indian Standards) and IRQS (Indian Register Quality System).

2. **ISO 9000 Certification:** ISO 9000 certification is a contract between independent parties. It is basically a set of guidelines. It is a series of three standards which are as follows:

- ISO 9001: It is used in software development organizations. It is applicable to organizations which are engaged in design, development and servicing of goods.
- ISO 9002: It do not design products. It is involved in production only.
- ISO 9003: This standard is applicable to those organizations which are involved in installation and testing only.

3. **Reasons to get ISO 9000 Certifications:**

- If an organization have ISO 9000 certification, confidence of customers in that organization increases.
- This certification provides basic framework for the development of the software product.
- ISO 9000 certification points out the weak points of an organization and recommends some actions to the organization so that they can improve those weak points.
- This certification helps the organization to make development process more focused, effective, efficient and cost-effective. It increases the quality of a particular software product.

4. **Procedure to get ISO 9000 Certification**

- An organization who wants to get ISO 9000 certification, has to register itself to a registrar. It is known as application stage.

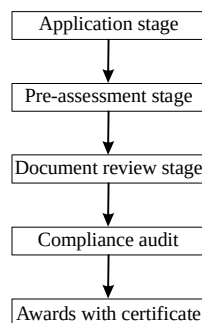


Fig. 3.18 Procedure to get ISO 9000 Certification

- After the registration, the registrar makes rough assessment of an organization. It is known as pre-assessment stage.
- In the third stage, the registrar reviews all the documents which are submitted by an organization during application stage. It is known as document review stage. The registrar suggests some possible improvements to the organization.
- During the fourth stage, which is known as compliance audit, registrar checks whether the organizations made those improvement or not which were suggested by him during document review stage.
- At the last stage, the registrar awards an organization with ISO 9000 certification after the successful implementation of all improvements suggested by the registrar.

After providing the ISO 9000 certification to the organization, the registrar monitors the organizations periodically to check whether the organizations maintain their quality or not. This process is known as continued surveillance.

5. **Issues related to ISO 9000 Certification**

- ISO 9000 certification does not provide any guidelines for an appropriate process and does not guarantees the process to be of high quality.
- There is no international accreditation agency exists which takes guarantee of ISO 9000 certification. Certification is awarded by non-governmental organization.

- After getting the certification, many organizations fail to maintain the quality.

II. SEI-CMM

1. It is also known as software engineering institute capability maturity model. It was proposed by SEI, USA in 1986.
2. It is a referenced model and used in two ways: capability evaluation and software process assessment.
3. It is intended to help software organizations to improve the maturity of their software processes. The focus is on identifying the key process areas.
4. SEI-CMM five maturity levels which are as follows:

- **Levels 1—Initial Level:** It is characterized by adhoc activities. It is also called chaotic level because different engineer follow their own process. No standard process is defined and followed during the software development. When the engineers leave the project, then it becomes very difficult for new engineer to understand the process.

- **Level 2—Repeatable:** At this level, size and cost estimation techniques like COCOMO, function point etc. are used to establish the tracking cost and schedule.

- **Level 3—Defined:** ISO 9000 aims to achieve this level. In this level, review techniques are emphasized to achieve phase containment of errors. Development and management activities are defined and documented.

- **Level 4—Managed:** At this level, process metrics and product metrics are collected and measured. Basically focus is on software metrics. Pareto charts, diagrams etc. are used to measure process and metric product metric.

- **Level 5—Optimizing:** For continuous process improvement, process and product measurement data are analyzed in detail. In this manner, organization finds out the best software engineering practices and innovations.

III. PSP—Personal Software Process

1. PSP is suitable for individual use. It is a scaled down version of industrial software process.
2. It is a framework that helps engineers to measure and improve the way they work.
3. It does not tell engineers how to analyze, design, code and test the software product. It just helps them to improve their work.
4. The methods of PSP are as follows:
 - (a) **Time measurement:** Use stopwatch to get an objective picture of the time spent.
 - (b) **PSP planning:** PSP planning helps individuals to plan their project.

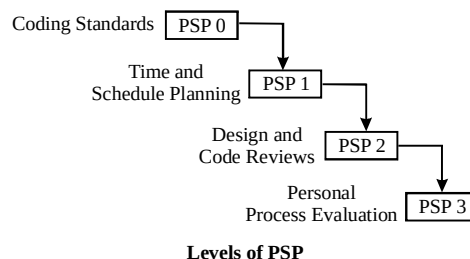


Fig. 3.19 Levels of PSP

IV. Six Sigma

1. The purpose of six sigma is to improve process to do things better, faster and at lower cost. It was pioneered by Motorola in 1995.
2. Six sigma is a rigorous and disciplined methodology that uses data and statistical analysis.

3. It improves the company's operational performance by identifying and eliminating defects from products.
3. Six sigma can be defined and understood at three distinct levels—metric, methodology and philosophy. It is applicable to every industry.

3.24 SOFTWARE QUALITY CONTROL (SQC)

1. Quality control is the pillar of quality assurance. Everything needs to have quality control from start to the finish.
2. Quality control is used to verify the quality of the output.
3. The aim of the quality control is to detect the defects and correct the defects.



4. Quality control is a Re-active technique. It is product oriented. The testing team members are responsible for quality control.
5. Activities of quality control are : Reviews, Testing etc.
6. In short, quality control is a subset of quality assurance.

3.25 DIFFERENCE BETWEEN QUALITY ASSURANCE AND QUALITY CONTROL

Sr. No.	Quality Assurance	Quality Control
1	Quality assurance is the process of managing quality.	Quality control is used to verify the quality.
2	Quality assurance is a pro-active technique. It identifies weakness in the process	Quality control is a Re-active technique. It identifies the defects and corrects the defects.
3	The aims of the quality assurance is to prevent introducing defects in the software application.	The aim of the quality control is to detect the defects in the software application.
4	It is process oriented.	It is product oriented.
5	It does not involve executing the program.	It involves executing the program
6	Example—verification	Example—validation or software testing.
7	It is done before quality control.	It is done after quality assurance.
8	It is a staff function.	It is a line function
9	All people who are involved in the software development process are responsible for quality assurance.	Testing team members are responsible for quality control.
10	It is a set of activities for ensuring quality in software engineering processes.	It is a set of activities for ensuring quality in software products.
11	Activities are—Audits, Training, Process definition and Implementation.	Activities are—Reviews, Testing
12	It is a subset of software test life cycle (STLC).	It is a subset of quality assurance (QA).

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 4: SOFTWARE REQUIREMENT ANALYSIS

4.1 INTRODUCTION

4.2 SYSTEM ANALYSIS

4.2.1 ROLE OF SYSTEM ANALYST IN SOFTWARE DEVELOPMENT PROCESS

4.3 REQUIREMENT GATHERING AND ANALYSIS

4.4 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

4.4.1 NEED OF SRS DOCUMENT

4.4.2 ATTRIBUTES/CHARACTERISTICS OF SRS DOCUMENT

4.4.3 STRUCTURE/ORGANIZATION OF SRS

4.4.4 REASONS OF BAD SRS DOCUMENT

4.5 STRUCTURED ANALYSIS

4.5.1 OBJECTIVES OF STRUCTURED ANALYSIS

4.5.2 PRINCIPLES OF STRUCTURED ANALYSIS

4.6 TOOLS FOR STRUCTURED ANALYSIS

4.1 INTRODUCTION

The software development must be based on the requirements raised by the customer. The requirements must reflect the user's view in both for designing and implementation. The documentation of user requirements is done in a systematic in the form of a document SRS. It is written in a natural language, and contains a description of what the system will do without describing how it will do it. It is designed to describe user requirements, system requirements, limitation, functions, economic and social issues. The following section describes requirement analysis, need of SRS, format of SRS and detail about different tools.

4.2 SYSTEM ANALYSIS

1. System analysis is a planned and systematic investigation to determine the functions of a particular system.
2. Its aim is to study the application area and its problems. For this purpose, study involves many data gathering methods like interviews, consultations and observations etc.
3. A person called system analyst conducts this study and identifies various activities, objectives and procedures to achieve his goal.

4.2.1 Role of System Analyst in Software Development Process

System analyst is an engineer who gathers and analyzes the customer requirements. System analyst gathers data and analyzes the gathered data to understand the customer's requirements. The role of system analyst is as follows:

1. System analyst defines problem in a precise and cleared manner.
2. System analyst uses many data gathering methods like interviews, consultations and observations.
3. After the data collection, system analyst analyzes the collected data to get the correct solution of a problem.
4. System analyst also studies the pros and cons of various proposed solutions and choose the one best solution for a problem.
5. System analyst develops plans, design those plans with standards to achieve goals.
6. System analyst visit to customer and make SRS.

4.3 REQUIREMENT GATHERING AND ANALYSIS

1. Many projects are failed to satisfy customers. This is because developers started developing the software without clear understanding of what the customers exactly wanted.

2. Whether a developer uses his/her complete knowledge to develop software, but it is impossible to develop a satisfactory solution without a clear understanding of a problem. If a developer fails to understand the customer's requirement, he/she is unable to develop satisfactory software.

3. Today, in many organizations, developers start spending considerable time to understand the exact requirements of the customer. This is known as requirement analysis and specification phase.

4. The aim of the requirement analysis and specification phase is to fully understand the customer requirements and to systematically document these specifications. SRS (software requirement specification) document is the final outcome of the requirement analysis and specification phase.

5. The main activities performed during the requirement analysis and specification phase are:

- (I) Requirement Gathering
- (II) Requirement Analysis
- (III) Requirement Specification
- (IV) Requirement Validation

(I) **Requirement Gathering:** It is also known as requirement elicitation.

1. In the requirement gathering activity, system analyst collects the data. It sounds very simple but it is very difficult process. It is very difficult to collect the relevant and useful information from large number of people.
2. For this purpose, working model is used. This working model helps the system analyst to collect the customer's requirements. But in the absence of working model, system analyst uses his/her experience, imagination and creativity for collecting customer's requirements.
3. System analyst uses various methods/ways to collect the customer's requirements which are as follows:

(i) **Interview:** Interview is an effective data collection method. It is basically consist of asking questions, listening to individuals and recording their responses. In this method, one is

interviewer and one is interviewee. Interviewer is one who takes interview and interviewee is one who gives interview. It can be conducted individually or as a group.

The following are the types of interview:

— **Informational Interview:** The objective of this interview is to ask for advice and gain more knowledge about a particular field. The knowledge that we gain here will be sharper and more informed.

— **Telephonic Interview:** A phone interview is a very cost effective way to collection information. It can last anywhere from 10 to 30 minutes. It is very challenging because interviewer can't see body language of interviewee.

— **Face to Face Interview:** It is sometimes known as individual interview. It is based on evaluation method, individual youth, volunteers or parents. The number of interviews and selection of interviewees will depend upon the purpose, time and resources.

Advantages of Interview:

- It provides deep and free response.
- It is flexible and adaptable method.
- It glimpse into respondent's tone, gestures.
- It has ability to probe, follow-up, clarify misunderstanding about questions

Disadvantages of Interview:

- It is costly in time and personnel.
- Sometimes, it become impractical with large numbers of respondents.
- It requires skill among interviewer and interviewee.
- It may be difficult to summarize responses.
- There may be chances of biasness among interviewer, respondent, situation

(ii) **Questionnaire:** In this type of data collection method, questionnaire is made which consists of multiple questions. These questionnaires are sent to people via mail, courier or by hand. This is a useful technique because no one knows your intention and body language. But it is the time consuming process.

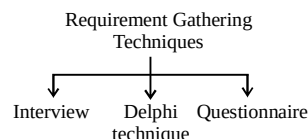


Fig. 4.1 Requirement Gathering Techniques

(iii) **Delphi technique:** The Delphi technique is a reliable and creative method. It begins with the development of a set of open-ended questions on a specific issue. These questions are then distributed to various experts. The responses to these questions are summarized and a second set of questions that seek to clarify areas of agreement and disagreement is formulated and distributed to the same group of experts.

Advantages of Delphi Technique:

1. It is conducted in writing and does not require face-to-face meetings.
2. It helps to keep attention directly on the issue.
3. It is inexpensive.

Disadvantages of Delphi Technique:

1. Its information comes from a selected group of people and may not be representative.

2. Its tendency is to eliminate extreme positions.
3. It is a time-consuming technique.
4. It requires skill in written communication.
5. It requires adequate time and participant commitment.

(II) Requirement Analysis:

1. The aim of requirement analysis activity is to analyze the gathered data and to obtain the clear understanding of the software product which is going to be developed.
2. The purpose of this activity is to fully understand the exact requirements of the customer and remove all ambiguities, incompleteness and inconsistencies in the requirements.
3. With the help of requirement analysis activity, the system analyst is able to clearly understand the following basic questions:
 - What is the problem?
 - What are the exactly data input to the system?
 - What are the exactly data output to the system?
 - What are the possible procedures?
 - What are the possible solutions?
 - Why is it important to solve the problem?



Fig. 4.2 Steps of Requirement Gathering and Analysis Process

(III) Requirement Specification

1. During the software requirement specification activity, analyzed information is translated into document form which defines the set of requirements. This document is known as software requirement specification.
2. The document is basically an agreement or contract between the customer and suppliers on what the software product is going to do.
3. The document is prepared by the senior analyst (SA).
4. The document specifies two types of requirements i.e. user requirements and system requirements.

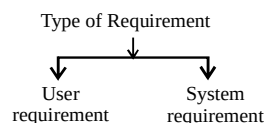


Fig. 4.3 Type of Requirement

User requirements are detailed description of customer requirements. Whereas system requirements are detailed description of the functionality to be provided by the system.

(IV) Requirement Validation:

1. In the requirement validation activity, it is checked that whether the requirements mentioned in the requirement specification are complete and consistent or not.
2. In this activity, errors are discovered in the requirement documentation. Requirement documentation is modified to remove the discovered errors.

4.4 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

1. SRS is prepared by the senior analyst (SA).
2. SRS is basically an agreement between the customer and suppliers on what the software product is going to do.
3. It is the final outcome of requirement gathering and analysis phase.
4. It is a complete description of behavior. It describes the scope of the product.
5. SRS document specifies the certain functions of a particular software product in a specific environment.
6. SRS document is very difficult to write. It covers all the requirements of the customer, functional and non-functional capabilities of the software product.
7. All functions/tasks of software development process are based on the software requirement specification (SRS).
8. Users of SRS document: Many people use SRS document according to their needs. The users of SRS document are: customers, software developers, test engineers, maintenance engineers etc. Software developers use SRS document to ensure that they are developing the software product according to the customer's requirements. Maintenance engineers use SRS document to understand the functionality of the software product.

4.4.1 Need of SRS document

1. SRS removes the communication gap between customer and supplier.
2. SRS document helps the client's/customer's to fully understand his/her own needs along with the developer's capabilities. It forces client to think and visualize his/her needs and discuss with developer in detail.
3. SRS document almost guarantees that after the development of software product, the client will be happy with this project.
4. With the help of SRS document, the client can determine whether the software product meets his/her requirements or not.
5. If a good quality SRS document is used in the software development process, then the developed software product is of high quality and low cost. The software can be developed and delivered in a fixed duration.

4.4.2 Attributes/characteristics of SRS document

The following are the attributes or characteristics of SRS document which enhances the quality of a software product.

1. SRS should be complete. The complete SRS specifies everything about the software product. The complete SRS includes a table of contents, page number, figure numbers etc.
2. SRS should be modifiable in easy manner when requirements are changed.

3. SRS should be unambiguous. Single requirement should have only one interpretation. Single requirement with many interpretation leads to confusion state and developed unsatisfactory software product.
4. SRS should be precise. It is used as a blue print for the coding in a program. It should also follow writing standards. SRS should precisely defines the system's capabilities along with interfaces.
5. SRS should be accurate. It should specifies exact requirements of the customer and system. The correct SRS clearly explains the functional and non-functional capabilities of the system.
6. SRS should be testable. The good SRS document easily identifies and removes its errors.
7. SRS should be traceable. Each requirement in SRS document should be traceable from source to implementation. It should also be able to work in backward direction means from implementation to source in necessary conditions.
8. SRS document should be consistent. Its capabilities, functions and performance levels should compatible and consistent. An SRS should not suffer with conflicting terms, conflicting characteristics and contradictory specifications.
9. SRS should be verifiable. Every single requirement in the document should be verifiable against the developed software product. It removes the communication gap between client and supplier.
10. SRS should be valid. It should be written in natural language so that all participants can easily understand, analyze and accept it.
11. SRS should be well structured. A well-structured SRS document is easy to understand and modify.

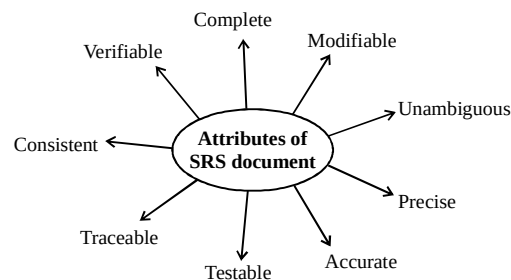


Fig. 4.4 Attributes of SRS Document

4.4.3 Structure/Organization of SRS

The SRS document should be organized in proper sections. The following is the structure of SRS document.

1.	Introduction
a.	Purpose
b.	Overview
c.	Environmental Characteristics
•	Hardware
•	Peripheral
•	People
d.	Definitions, Acronyms and Abbreviations
2.	Goal of Implementation
3.	Functional Requirements
a.	User Class I
•	Functional Requirement
•	Functional Requirements
b.	User Class II
•	Functional Requirements
•	Functional Requirements and so on
4.	Non-Functional Requirements
a.	External Interface
(i)	User Interface
(ii)	Software Interface
(iii)	Communication Interface
b.	Constraints
c.	Performance Requirements.
5.	Behavioural Description
a.	System States
b.	Events and Actions

Fig. 4.5 Structure of SRS

The following are the explanation of structure of SRS document. These sections are also known as components of SRS document.

1. **Introduction:** It explains the context in which the system is being developed. It provides the overall description and environment characteristics of the system. The environmental characteristics specifies the hardware (central processing unit usage, memory usage, network communication etc.) on which the system will run. It also explains the people who are involved in the development and usage of the system. This section also explains the definitions, acronyms and abbreviations such as QoS: Quality of Service. It meets what users actually want and maximize their utility, GUI: Graphical user interface, CPU: Central processing unit of a computer etc.

2. **Goal of Implementation:** It offers some general suggestions regarding software development. It provides guidelines for the successful development of software product. It is not tested by the user at the time of acceptance testing.

3. **Functional Requirements:** It is the critical part of the SRS. It explains all the functionalities required by the user. It specifies which output should be produced from the given input. All the operations to be performed on input data is specified in this document. It describes the relationship between input and output of the system. For each functional requirement, a detailed description of all the inputs, their resources, unit of measure and the range of valid inputs must be specified.

4. **Non-Functional Requirements:** This section explains the characteristics of the system such as maintainability, portability, throughput, usability etc. It explains the external interface

(user interface, software interface and communication interface) of the system. It also specifies the constraints which restrict the choice of design like security, reliability and fault tolerance. All the requirements related to performance are clearly specified. The performance requirements are of two types:

- **Static requirements:** These requirements do not impose constraints on the execution behaviour of the system.

- **Dynamic requirements:** These requirements specify constraints on the execution behaviour of the system.

5. **Behavioural Description:** It is used to specify the possible states of the system and the transitions among these states due to occurrence of the system. It is not necessary for all the system.

4.4.4 Reasons of Bad SRS Document

The following are the reasons of bad SRS document:

1. **Ambiguous:** The SRS document specifies an ambiguity in the requirement. It arises many interpretations of that requirement.
2. **Incomplete:** Some requirements have been overlooked and the document left many important aspects of the requirement.
3. **Inconsistent:** The requirements in SRS document contradict each other.
4. **Unstructured:** The document is not in structured manner and makes confusion among developers and users.
5. **Over specification:** It restricts the freedom of the designer in arriving at good design.
6. **Wishful thinking:** When the document specifies those requirements which are very difficult to achieve.

4.5 STRUCTURED ANALYSIS

1. Structured analysis is the activity of deriving a structured model of the system requirements.
2. Structured analysis techniques help an analyst to decide what type of information is obtained at different points in analysis.
3. It helps to organize information in such a manner that analyst is not over-burdened with complex problems.
4. It is basically a technique/methodology which is used to build a system model which explains the flow and content of information.
5. Special notations and symbols are used to describe the system functionality.

4.5.1 Objectives of Structured Analysis

1. The aim of the structured analysis is to define a set of requirements that can be validated once the software is built.
2. It describes the requirements of the customer.
3. It establishes a base for the software design creation.
4. It is basically a documentation for the system which describes the clear and complete specification.

4.5.2 Principles of Structured Analysis

The structured analysis approach follows the following principles:

1. The behavior of the system must be represented.
2. The functions that the software is to perform must be defined.
3. The analysis process should move from essential information towards implementation detail.
4. The model should be partitioned in such a way that details should show in a hierarchical manner.
5. The problem of the software must be understood and represented.

4.6 TOOLS FOR STRUCTURED ANALYSIS

The following are the tools used for structured analysis:

- (I) Data Flow Diagram (DFD)
- (II) Data Dictionary (DD)
- (III) Process Specification (P-Spec)
- (IV) State Transition Diagram (STD)
- (V) Entity Relationship Diagram (ERD)

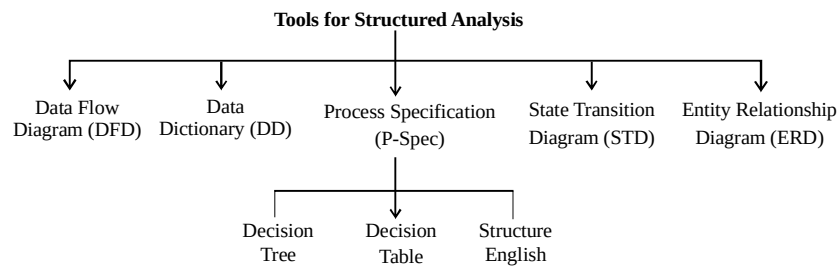


Fig. 4.6 Structured Analysis Tools

I. Data Flow Diagram (DFD):

1. It was developed by Larry Constantine. It is also known as bubble chart.
2. It is a graphic representation of the flow of data or information through a system
3. It represents the logical data flow rather than how they are processed. It only show the flow of data through the system.
4. It consists of a series of bubbles joined by lines. The bubbles represent data transformations and lines represent data flow.
5. DFD can be partitioned into levels. Each level has more information flow and data functional details than the previous level.
6. **Symbols of the DFD:** There are different types of symbols used to construct DFDs. The meaning of each symbol is explained below:

Sr. No.	Symbol	Meaning
1.		Source or destination of data
2.		Process which transforms data flow
3.		Data flow
4.		Data Store

Table 4.1

7. Rules for the construction of DFD:

- (i) The direction of flow of data is from top to bottom and from left to right.
- (ii) The names of sources, data stores and destinations are written in capital letters.
- (iii) The processes are named and numbered for easy reference. When a process is exploded into lower level details, it is also numbered.

8. Level of DFD:

(i) **Level 0:** It is also known as context level DFDs or highest abstraction level DFDs. It depicts the entire information system as one diagram covering all the essential details. It represents scope of the system and identifies external entities and related inputs and outputs

(ii) **Level 1:** It depicts basic modules and flow of data among various modules The Level 0 DFD is broken down into more specific, Level 1 DFD. It provides overview of full system. It identifies major processes and data stores.

(iii) **Level 2:** At this level, DFD shows how data flows inside the modules mentioned in Level 1. It describes the deeper level of understanding unless the desired level of specification is achieved. Level 1 process is expanded into more detail.

Advantages of DFD:

1. Data flow diagrams are easy to understand check and change data.
2. It give a very clear and simple look at the organization of the interfaces between an application and the people or other applications that use it.
3. It is an effective and efficient method in the absence of required design.
4. It is not limited to software.

Disadvantages of DFD:

1. Modification to a data layout in data flow diagram may cause the entire layout to be changed.
2. Maintenance of data flow diagram is harder, more costly and error prone because in the large application, number of units in a DFD are high and changes are impractical to be made on DFDs.

II. Data Dictionary (DD):

1. It is also known as Meta data. A data dictionary is a structured repository of data about data.
2. It is a centralized collection of information.
3. It stores meaning and origin of data, its relationship with other data, data format for usage etc.
4. It removes any chance of ambiguity and helps to synchronize the working of programmers and designers.
5. It is an important and essential step in building a database.
6. Data dictionary contains following information:
 - (i) **Data Element:** It is a smallest unit of data which has no further decomposition. It provides the description of data like: name, source, date of origin, users etc.
 - (ii) **Data Structure:** A data structure is a systematic way of organizing and accessing data. It explains how data are stored in a computer.
 - (iii) **Data Flow:** Data flows are data structures in motion.
 - (iv) **Data Stores:** Data flows are data structures at rest. It includes files and tables which stores the information from where the data enters into the system and exists out of the system.

III. Process Specification (P-Spec):

1. It is used to define all the processes which appear in the DFD.
2. All bottom level processes are specified.
3. The tools of process specification are:

Decision Tree	• It is a way of breaking down the complicated situation into easier to understand scenario. • Graphical technique representing decisions using a series of nodes and branches • Each node is a decision point - a choice has to be made • Each branch has a corresponding value to the decision choice • Subsequent action is the result. • It is easy to understand, no need for special training.
Decision Table	• Representation of logic that is part of the processing • Based on a set of conditions, different actions will be performed • Can be simplified by removing impossible actions • Used when the process result is based around several different variables and the logic gets too complex for other methods
Structure English	• In Structured English, decisions are made through IF, THEN and ELSE statements. • It can use CASE construction as well as LOOP construction. • It is basically a subset of English with restrictions on usage. • It can use some acceptable verbs such as FIND, ADD, SUBTRACT, MULTIPLY, DELETE etc.

Table : 4.2

IV. State Transition Diagram (STD):

- It represent a process specification for a control bubble in DFD.

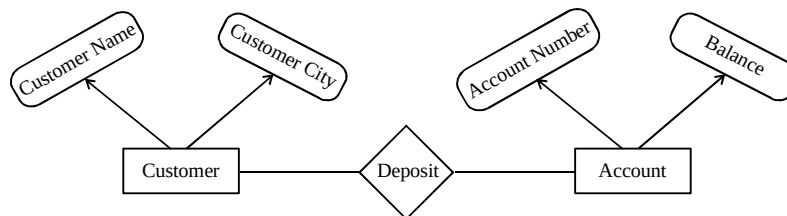
- It is a tool for representing the design.
- It has one initial state and may have multiple final states.
- **State transition diagrams have 4 components :**

States	It is some behavior of a system that is observable and that lasts for some period of time like doing, waiting etc.
Transitions	It is an instantaneous change in state/behavior.
Conditions	A condition is typically some kind of event. It is an event in the external environment which triggers a transition to a new state.
Actions	An action is the appropriate output or response to the event. It is a response sent back to the external environment whose result is stored by the system.

Table : 4.3

V. Entity Relationship Diagram (ERD)

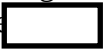
1. Entity relationship diagram is based on real world. It is a collection of entities (objects) and relationship among entities.
2. Entity relationship model has mainly three components:
 - (i) Entity
 - (ii) Attributes
 - (iii) Relationship



E-R Model

Fig.4.7 E-R Model

(i) Entity:

- Entity is a person, place or thing which can be identifies.
- It is represented by rectangle 
- It is of two types:

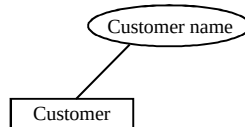
Weak entity: It depends upon some other entity. It is represented by 

Strong entity: It is not dependent on some other entity. It is represented by 

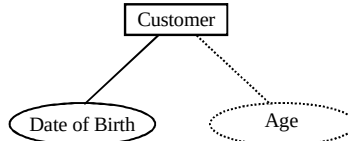
(ii) Attributes:

- Attributes are the properties of entity.
- It is represented by ellipse 
- It is of four types:

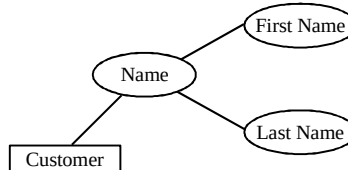
Single attribute: These attributes cannot divided into subparts.



Derived attribute: The value of these attributes are derived from another attribute.



Multivalued attribute: These attributes have more than one value.



Composite attribute: The value of these attributes can be divided into another attributes.

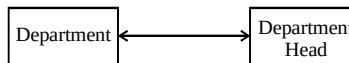
(iii) Relationship:

- It is used to connect entities. The entity involved in relationship is known as participants.

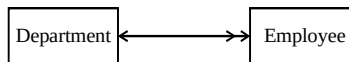
- The number of participants in a given relationship is known as degree of relationship.

- It is of four types:

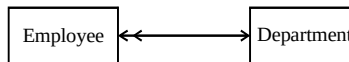
One to one relationship: In one to one relationship, for one record in entity A, there is exactly one record in entity B.



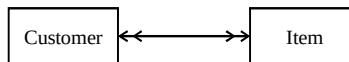
One to many relationship: In one to many relationship, for one record in entity A, there is more than one record in entity B.



Many to one relationship: In many to one relationship, for many records in entity A, there is only one record in entity B.



Many to many relationship: In many to many relationship, for many records in entity A, there are many records in entity B.



BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

SECTION- B

UNIT 5: SOFTWARE DESIGN

5.1 Introduction

5.1.1 Outcomes of a Design Process

5.1.2 Classification of Design Activities

5.1.3 Classification of Design Methodologies

5.1.3.1 Procedural Approach

5.1.3.2 Object Oriented Approach

5.2 Objective of Software Design

5.3 Criteria of Effective Software Design

5.4 Design Principles

5.5 Design Steps

5.6 Design Concepts

5.7 Design Quality Metrics

5.7.1 Cohesion

5.7.2 Coupling

5.7.2.1 Types of Coupling

5.8 Data Design

5.9 Architectural Design

5.10 Procedural Design (Function Oriented Design)

5.11 Structured Design Methodology (SDM)

5.12 Design Verification

5.1 INTRODUCTION

Software design is a process of transforming the customer requirements into a form that is implementable by using a programming language. It is a very creative process. It is the basis of effective engineering. It should be practiced and learnt by experience and study of the already existing systems. The design phase begins when the requirements document is available. This is the first phase of

transforming the problem into a solution. The design process is made up of a set of principles, concepts and practices which help a software engineer to model the system or product that is to be developed.

According to the Stevens in 1991, “Software Design is the process of inventing and selecting programs that meet the objectives for a software system.”

In the words of Coad and Yourdon, “Software Design is the practice of taking a specification of externally observable behavior and adding details needed for actual computer system implementation, including human interaction, task management and data management detail.”

5.1.1 Outcomes of a Design Process

The Design Process is an activity in which software requirements are studied and analyzed to produce a description of the internal structure and organization of the system. During the software design phase, many serious, strategic, technical decisions are taken to achieve the required functional and quality requirements of a system. These decisions help in developing the software in a successful manner and carry out its maintenance in a well-defined manner to improve the quality of the Final Product. Software design transforms the problem into a solution. In this phase the customer requirements, the business requirements and the technical needs, all come together to formulate a product or a system.

5.1.2 Classification of Design Activities

A good software design can never be achieved by using a single step procedure but requires several steps. Software Design is an activity in which software requirements are studied analyze and then a systematic procedure is adopted to produce a description of the internal structure and organization of the system that becomes the basis of its construction. Broadly speaking thus are two activities.

1. Software Architectural Design
2. Software Implementation Design

The meaning and scope of these two activities tend to vary considerably. In software architectural design, the top level structure and organization of the system is described and different components are identified. The system is decomposed and organized into components and interfaces between these components are described. The outcome of this architectural design is known as Program Structure or Software Architecture.

In the second activity i.e. the software implementation design, the data structure and the algorithms of different modules are designed and its outcome is usually known as Module Specification Document. In this activity each component is sufficiently described to allow for its coding. After studying different design approaches; seven steps are necessary to obtain a good design these steps are as follows:

(i) Function decomposition, (ii) Interface definition, (iii) Operational time line development, (iv) Data Definition, (v) Concurrency and real time consideration, (vi) Consolidation, (vii) Test procedure development.

5.1.3 Classification of Design Methodologies

Software Design is the central activity of software engineering. It is also the integrative activity at the core of software engineering. The designer should be methodical, disciplined, communicative and self-analytical. Various design methods are used today. The design principles and concepts lay a foundation for the creation of a Design Model that encircles around representation of Data, architecture, interface and components. The major software Design Methods include – (i) Function Oriented Design Methods, (ii) Data structure leased Design methods, (iii) Object oriented Design Methods, (iv) Reuse Based Design Methods.

Design Methodology is a very broader area and it is a problem solving activity. Design Methodology focuses on:

1. **Divergence:** It applies critical thinking through qualitative and quantitative approach to create better designs solutions.

2. **Transformation:** It includes better redefining of specifications of design solutions that leads to better guidelines for traditional and contemporary design activities.
3. **Convergence:** Prototypes probable scenarios for better design solutions.
4. **Sustainability:** It controls systematically the process of exploring, redefining and prototyping of designs solutions continually overtime.
5. **Articulation:** It articulates the visual relationship between the parts and the whole.

The true Goal of Design Methodology is to gain key insights to obtain more holistic solutions to achieve better experiences for users with products, services, environments and systems.

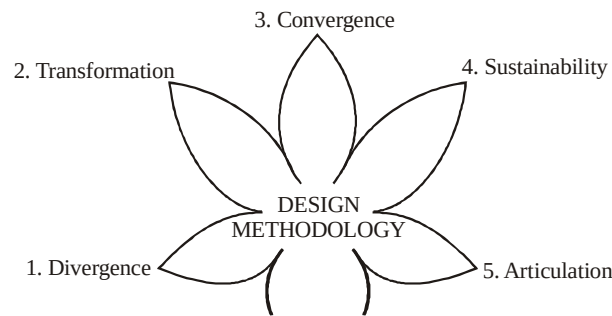


Fig. 5.1: Software Design Methodology

5.1.3.1 Procedural Approach

Procedural design techniques are very common and popular and at present they are being used in many software organizations. It is the result of focusing full orientation to the function of the program. This is an approach to software design where the design is decomposed into a set of interacting units where each unit of module performs a specific function Niklans Wirth who is the creator of Pascal and a number of other languages are one of the best known advocates of this method. His special variety to known as step-wise refinement. This stepwise refinement is a top-down strategy where a program is refined as a hierarchy of increasing levels of detail (Mills 1988). Refinement is actually a process of elaboration.

Procedural design is also named as component level design. It occurs after data, architectural and inter face designs have been developed. The aim is to translate the design model into an operational software.

5.1.3.2 Object Oriented Approach

Object-oriented design is a design strategy of planning a system of interacting objects for the purpose of solving a reaching a predefined software generated result. These Objects manage their own private state and offer services to other objects and hide representation of the state to limit access to it.

Object-oriented concepts

The five basic concepts of object-oriented design are:-

1. Object/Class
2. Information hiding
3. Inheritance
4. Interface (object-oriented programming)
5. Polymorphism

5.2 OBJECTIVE OF SOFTWARE DESIGN

Software Design is expected to deliver the requirement as specified in the Feasibility Report. The quality of the design is assessed with a series of technical reviews or design walk throughs. The main software objectives which a software design should provide are as follows:

Basic Objectives

1. To produce such models that can be analysed and evaluated to find whether they allow the various requirements to be fulfilled.
2. To study and evaluate different alternative solutions and trade-offs.
3. To plan the following development activities.

Major Objectives

1. To understand the requirements and to generate the solution to accomplish that requirements.
2. To transform the problem space of the implementation.
3. To solve complex problems by dividing them into a set of sub-problems and then finding partial solutions easily in a better way.
4. To identify the proper level of detail when design should stop and implementation should start.
5. To define the software architecture and describe it through step by step details.
6. To make the module functionally independent.
7. To implement all the explicit requirements of the Analysis Model.
8. To define the relationship between major structural element of the software.
9. To produce a design that is readable and understandable.
10. To provide a complete picture of the software that can be assessed for quality from an implementation prospective.
11. To translate the customers requirements into a well-furnished software product or system.
12. To follow a simple design approach so that designs are easily understood, easily built and easily tested.
13. To accommodate the changes that may be required during its life time.

5.3 CRITERIA OF EFFECTIVE SOFTWARE DESIGN

Most researchers and software engineers agree on certain characteristics that every Good Software Design must possess. The Goodness of a design depends on the targeted application. The definition of a Good Software Design can vary depending on the application on which it is based. The characteristics of a Good and effective Software Design are as under:

1. Correctness—A Good Design must implement correctly all the functionalities of the system.
2. A Good design should be easy to understand.
3. It should be efficient and should accommodate all the implicit requirements desired by the user.
4. It should not be difficult to change.
5. A good design should exhibit an architectural structure with recognizable design patterns.
6. An effective software design should be modular. It should be logically divided into elements that perform specific functions and sub functions.
7. A good design always possesses well written documents.
8. A design is said to be good and effective if it separates data, procedures and tuning considerations as far as possible.
9. In a good design, every requirement is testable. If a design cannot be tested easily against its requirements, then it is unacceptable.
10. Each and every point about the design should be easily communicated to all concerned through proper abstractions and representations.
11. A good design is liable to be repeated or re-used.

Each of the above characteristics is actually a goal of the design process. The design should be readable, understandable and must provide a complete picture of the software and it should fulfill the requirements desired by the customer.

5.4 DESIGN PRINCIPLES

Fundamental Principles for any discipline remain the same throughout. They provide the underline basis for development and evaluation of techniques. Fundamental concepts of software design include abstraction, structure, information hiding, modularity, concurrency, verification, testability, discreteness, and reusability and design aesthetics.

Some factors that help in making good software are creative skill, past experience and an overall commitment to quality. Davis has suggested a set of principles for software design which have been adapted and extended in the following list :

1. The Designer Process should not suffer from tunnel “Vision”. A Good designer applies alternative approaches judging each according to the requirements of the problem and also takes into consideration the resources available to do the job.
2. The design should be traceable to the analysis model — A Good design must have means to track how the requirements have been satisfied.
3. The design should not re-invent the wheel — A set of design patterns should be chosen as an alternative to re invention. Design time should be used to represent truly new ideas and integrate all preexisting patterns.
4. The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world.
5. The design should exhibit uniformity and integration.
6. The design should be structured to accommodate change.
7. The design should be structured to degrade gently, even when data, events or operating conditions are encountered. It should be designed to adjust according to the unusual circumstances.
8. Design is not coding and coding is not design — even when detailed procedural designs are produced, the level of abstraction of the design model is higher than source code design decisions at the coding level address the small implementation details which enables the procedural design to be coded.
9. The design should be assessed for quality as if being created not after the fact.
10. The design should be reviewed to minimize conceptual errors.

By following the design principles, Software Engineer produces a design of high quality which shows both external and internal quality factors. The user readily observes the external qualities (Speed, correctness, usability and reliability etc.). The internal quality factors are of immense significance to the Software Engineer. The designer must understand basic principles to create a high quality design from the technical perspective.

5.5 DESIGN STEPS

By critically examining and evaluating different design approaches, seven steps have been devised which should be followed to obtain a good design. These steps are as follows:

1. **Function Decomposition:** In the step software product is partitioned into smaller components. Rules are observed to assure that recombining and reassembling the components to intelligent testing scenarios which helps the management in maintaining control over the process. Block diagrams, structure charts and tools are the software development tools. Some characteristics and constraints for partitioning are as follows:

- (a) Unit Testing Improvement
- (b) System testing enhancement
- (c) Cohesion and coupling
- (d) Managerial Visibility
- (e) Managerial control
- (f) Information hiding
- (g) Contractual and organizational compatibility
- (h) Data, time dependency and function separation within limits.

The components are the building blocks. The partitioning step is the realization of a structure or architecture for the final design of the software product.

2. **Interface Definition:** Component to component and component to external interface are identified and defined. Many tools like object diagrams, data flow diagrams, timing chart and data dictionaries support this step. Another requirement of this step is to provide interface information that a designer, coder and tester needs for using other design components in developing a software product.

3. **Operational time line Development:** In this step realistic operational timeline or scenario to exercise are developed for each design step. These timelines should have operator interactions with the system.

4. **Data Definition:** This step is linked to develop data structures and this step serves the purpose of defining the files, records, fields, access methods, global and local data structure and the procedure to maintain files. Transactions are chalked out and the file size estimates are made. Tools for this step are data dictionary, information modeling and entity relation diagrams.

5. **Concurrency and Real Time Considerations:** The fifth step is linked to concurrence and real-time requirement. It is strongly desired to decouple timing requirements and their design and correctness tests from function, performance and data correctness. This step is used to establish the timing for the design component. The tools used for this step are timeline extensions to data flow diagrams and McCabe's analysis approach.

6. **Consolidation:** Information gathered from in steps one to five and a realization of the consolidated design information is used as a direction to coders in the form of diagrams, pseudo code structured English or any other form of program design languages.

7. **Test procedure Development:** This step studies those ways and means by which the design can be tested as a design component and finds how the design component contributes to confirm a system level acceptance criteria. This slip helps in verification that all requirements in the software requirements specifications have been met and traces if to implementation in the system design specifications.

Control hierarchy and data structure are too important aspects of the overall software products Architecture.

Control Hierarchy: The control relationship among various components of the product is sometime, referred to as visible. Each component has access to many other components and connectivity is done only to those components directly involved by the given component.

Data Structure: It is a representation of the relationship among data items in a software product design organization, access methods and controls, definitions, degree of association etc.

Data structure is consistent with overall architecture and if stays connected with control and procedure design throughout the design process. Formal design reviews can be inserted at proper interval to ensure that design efforts are fully coordinated.

5.6 DESIGN CONCEPTS

By following a rigorous approach in any engineering activity, satisfactory products can be produced. We can control. The costs of the products and increase their reliability. This does not mean to constrain creativity rather it improves the engineer's confidence in precision and accuracy. He brings creative results after critically analyzing in the light of a rigorous assessment.

Some basic steps have been devised over the past four decades which provides a basis to the software designer to apply more sophisticated design methods. He becomes capable of knowing different criteria to partition software into individual components; he comes to know how data structure detail can be separated from a conceptual representation of the software.

In the words of M.A. Jackson, "The beginning of Wisdom for a software engineer is to recognize the difference between getting a program to work and getting it right". The basic concepts for the Software Design are given below:

1. Abstraction

2. Refinement
3. Modularity
4. Software Architecture
5. Control Hierarchy
6. Structural Partitioning
7. Data Structure
8. Software Procedure
9. Information Hiding

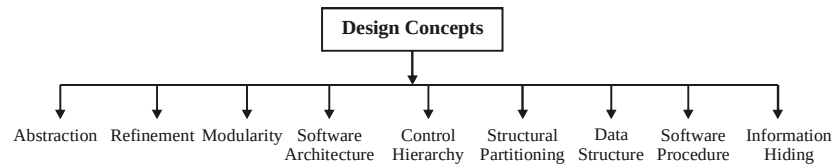


Fig. 5.2: Design Concepts

1. **Abstraction:** It is a very powerful concept and an intellectual tool used in all engineering disciplines which permits a designer to consider a component at an abstract level without worrying about the details of the implementation of the component. It is a special case of separation of concerns where in concern of the important aspect is separated from the concern of the unimportant details. Abstraction is an indispensable part of the design process and is essential for problem partitioning. Abstraction is used to existing components as well as components that are being designed. Abstraction of existing components plays a vital role in the maintenance of the system and we become able to modify the system in a step wise manner. Abstraction reduces the information content of a concept or an observable phenomenon, typically to retain relevant information for a particular purpose. Abstracting happiness to an emotional state of mind reduces the amount of information conveyed about the emotional state. There are different levels of abstractions. These levels are as follows:

- (i) **Data Abstraction:** In Data Abstraction; the component is specified by the data types or data object and operations that performed on data object. Like any data object, the data abstraction for door would comprise of a set of characteristics (e.g. door type, swing direction, opening mechanism, weight, length, breadth, height etc.) Many modern programming languages provide mechanisms for creating abstract data types.
- (ii) **Procedural Abstraction:** It is a named sequence of instructions that has a specific and limited function. In procedural abstraction component is specified by the function it performs e.g. Auto sum function in Excel. OR the word open for a door that implies a long series of procedural steps (i.e. walk to the door, reach out and grasp knob, turn knob and pull door, step away from moving door etc.)
- (iii) **Control Abstraction:** It means a program control mechanism without specifying internal details. It describes the external behavior of that component. In example of a control abstraction is the Synchronization Semaphore used to co-ordinate activities in an operating system.
- (iv) **Cluster Abstraction:** In cluster abstraction the component is in a group of related classes that work together. These are also called framework e.g. for net framework.

2. **Refinement:** Stepwise refinement is a top down strategy where in a program is refined in the form of hierarchy. In this technique decomposing of a system is done from high level specifications into more elementary levels. In each step, one part of high level description is taken and refined. It is actually a process of elaboration. Refinement is also known as “Stepwise Program Development and Successive Refinement” Niklaus Wirth suggested following activities:

- (i) Decomposing design decisions to elementary levels.
- (ii) Isolating design aspects those are not truly independent.
- (iii) Postponing decisions concerning representation details as long as possible.

- (iv) Carefully demonstrating that each successive step in the refinement process is a faithful expansion of previous steps.

The following steps should be revised at each level in the design:

- (a) Study and understand the problem.
- (b) Identify gross features of at least one possible solution.
- (c) Describe each abstraction used in the solution in some design description language.

Refinement begins with the specifications which are derived during requirements analysis and external design. It is a very effective method for describing small sized programs but fails in the large sized programs refinement helps the designer to know low-level details as design progresses.

3. **Modularity:** The real power of partitioning comes if a system is partitioned into modules, so that these modules are solvable and modifiable separately. It is a program unit. A module can be a macro, a function, a procedure a process or a package. Some criteria is followed to select modules so that modules support well defined abstractions e.g. coupling and cohesion are two modularization criteria used in Functional Abstraction.

Modularity promotes design clarity which cases implementation, debugging, testing, documenting and maintenance of the software product. A modular system has the following characteristics:

- (i) Each Processing Abstraction is a well-defined sub system which can be effectively used in other applications.
- (ii) Each function has a single well defined motto.
- (iii) Each function manipulates only one major data structure.
- (iv) Functions share global data selectively. It is convenient to identify all solutions that share a major data structure.
- (v) Functions that control instances of Abstract Data types are encapsulated with the data structure being manipulates.

4. **Software Architecture:** Shaw and Garlan (1996) suggested that Software Architecture is the initial step in producing a software design. They gave three design levels namely Architecture Design, Code Design and Executable design.

Software Architecture is the hierarchal structure of Program Components (Modules) and expresses the way in which these components interact and structure of data used by the components.

Architecture Design: It links the system capabilities observed in the requirements specification with the system components that will implement them. The architecture describes the inter connections among different modules. He also defines operators who create systems from sub systems.

Code Design: If involves algorithms and data structures the components such as programming languages primitives such as numbers, characters, pointers and control threads. There as primitive operators which involves the language's arithmetic and data manipulation primitives and composition mechanisms (arrays, files, procedures)

Executable Design: The code design is studied of a lower level of detail. It discusses memory allocation, data formats, bit patterns and so on.

Keeping all the above mentioned specifications the architectural design can be represented using any number of the following models in the figure of system I and system II, it is to be noted how one component is connected to the other only when the first component can invoke the other. For a given component, the set of other components to which arrows are drawn is called scope of control of the components. The components invoked by the given component are collectively referred to a scope of effect e.g. in System I-II, are two possible designs for the same system. It should be noted that no component should be in the scope of effect; if it is not in the scope of control. If the scope of the effect is wider than the scope of its control, it is almost impossible to give assurance that a change to the component will not damage the whole design.

Fan in: It is the number of components controlling a particular component.

Fan out: It is the number of components controlled by a component. Component “A” has a fan-out of 3 in system I but a fan-out 5 in system II.

Fan in for component ‘C’ is 1 in both systems.

Generally a system with high fan-out tries to minimize the number of components. The controlling component performs more than one function. Comparing in this way system I with low fan out may be better than System II.

Studying from another angle, when number of levels are increased a particular component is used more than once. If we design one general purpose component to do that task, the results design is more efficient and easier to test. In such cases design with a large number of levels is considered better one of the goals in designing systems is creating such components which have high fan in and low fan-out.

- (a) **Structural Models:** It puts forth Architecture as a systematized collection of program components.
- (b) **Frame Work Models:** These models enhance the level of design abstraction. If tries to identify architectural design frameworks that appear in similar types of applications.
- (c) **Dynamic Models:** They address the behavioral aspects by indicating how the system configuration may be changed as a function of external events.
- (d) **Process Models:** They lay emphasis on the design of the business or the technical process.
- (e) **Functional Models:** These models can be used to represent the functional hierarchy of a system.

5. **Control Hierarchy:** It represents the organization of program components and if shows a hierarchy of control. It does not represent procedural aspects of software. It does not apply to all architectural styles. Number of components under the control of a particular component can be represented diagrammatically as under.

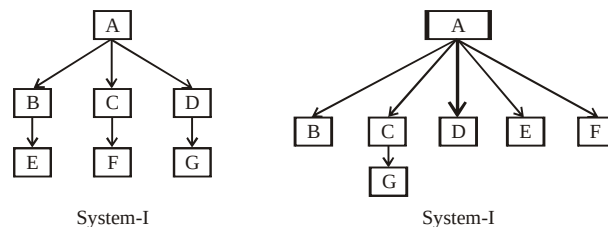


Fig. 5.3: One component controlling other components

6. **Structural Partitioning:** When the architectural style of a design follows a hierarchical nature, the program structure can be partitioned either horizontally or vertically.

Horizontal Partitioning: In this method the control modules (shown as shaded boxes) are used to co-ordinate communications between functions and execution of the functions. It has the following benefits:

- It results in software that is easier to test and maintain.
- It results in propagation of fewer side effects.
- It results in software that is easier to extend.

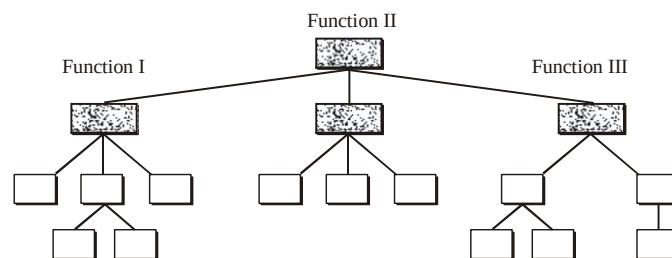


Fig. 5.4: Horizontal Partitioning

Disadvantage of Horizontal Partitioning: In this more data has to be passed across the module interface. This complicates the overall control flow of the problem, especially during speedy movement from one function to another.

Vertical Partitioning: This is also called factoring. In this the control (decision making) modules are located at the top and work is distributed in a top down manner in the program structure. Top level models perform control functions and do little processing. Modules that reside low in the structure perform all input, computation and output tasks.

The nature of change in the program structures requires vertical partitioning because a change in a control module (high in the structure) will have a higher probability of propagating side effects to modules that are subordinate to it. A change to a worker module (low level in the structure) is less likely to propagate side effects. Therefore vertically partitioned structures will be more maintainable a key quality factor.

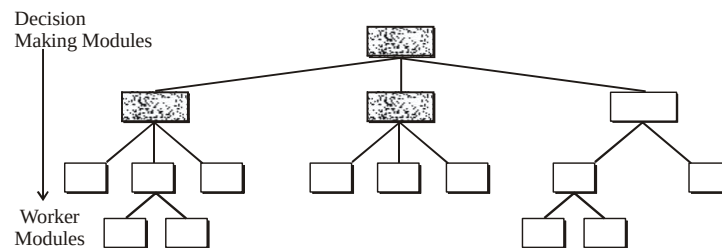


Fig. 5.5: Vertical Partitioning

7. **Data Structure:** Data structure is a representation of the logical relationship among individual elements of the data. As the structure of information has a great effect on the final procedural design. Data structure has an effect on the organization methods of access degree of associativity and processing alternatives for information. Although organization and complexity of a data structure depend on the thinking and skill of the designer, yet a limited number of classic data structures form the building blocks for more highly skilled and well-designed structures.

A scalar item represents a single element of information and it is the simplest form of data structure. Its access may be achieved by assigning a single address in the memory. The size and format of a scalar item are controlled by a programming language and they may vary within limits. A scalar item may be a logical entity of one bit length, an integer or floating point number being 8 to 64 bits long or a character string that is hundreds or thousands of bytes long.

When scalar items are well organized in the form of a list or a group, sequential vector is formed. Vectors open the door to variable indexing of information.

When the sequential vector is extended to two, three and finally an arbitrary number of dimensions, an n-dimensional space is created. In many programming languages, an n-dimensional space is called an array. Two dimensional matrix is the commonest n-dimensional space.

Items, vectors and spaces may be organized in various formats & linked list is a data structure that organizes noncontiguous scalar items, vectors or spaces in a manner (called nodes) so that they may be processed as a list. Each node contains the appropriate data organization and one or more pointers that indicate the address in storage of the next node in the list. Nodes may be added at any point in the list. It is done by redefining pointers to accommodate the new list entry. Other data structure is constructed using the fundamental data structures. A hierarchical structure is commonly encountered in applications that require information categorization and associativity. Data structure can be represented at different levels of abstraction. For example, a stock is a conceptual model of a data structure which can be implemented as a vector or a linked list. The internal working of a stock depends on the level of design detail.

8. **Software Procedure:** Program structure defines control hierarchy without taking into consideration sequence of processing and decisions whereas software procedure focuses on the processing details of each module individually. Procedure must provide a precise specification of processing including sequence of events, exact decision points, repeatable operations and even data organization and

structure. There is a relationship between structure and procedure. A procedural representation of software is layered. The processing indicated for each module must include a reference to all modules subordinate to the module being described as is clear from the diagram given below:

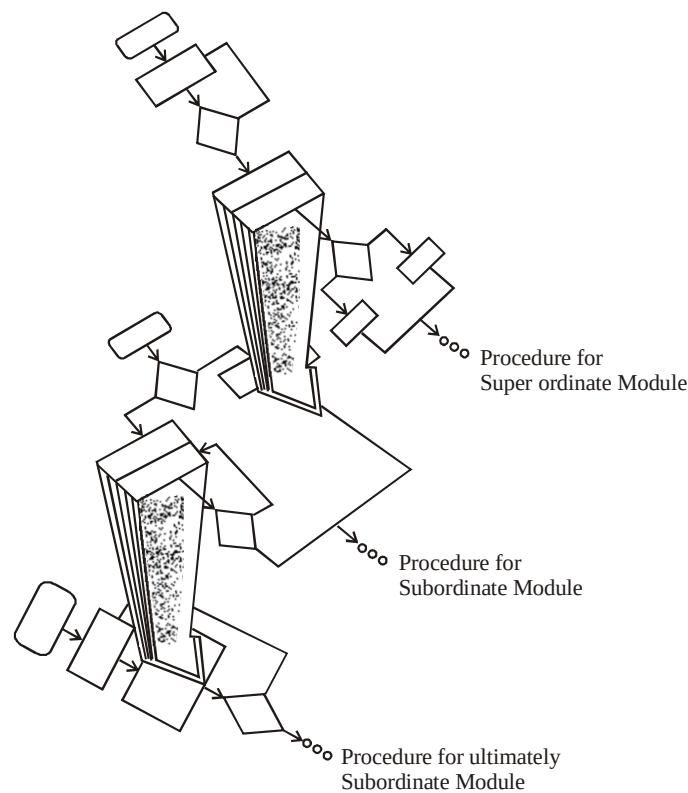


Fig. 5.6: Layered Software Procedure

9. **Information Hiding:** Modules should be designed and specified in such a way that each module in the system hides the internal details of its processing activities and modules may communicate through well-defined interfaces. The way of hiding in necessary details is known as Information Hiding. IEEE defines it as “the technique of encapsulating software design decisions in modules in such a way that the modules interfaces reveal as little as possible about the module’s inner workings; this each module is a ‘black box’ to the other modules in the system.”

According to Parnas, design should begin with a list of difficult design decisions and design decisions that are likely to change. Each module is designed to hide difficult and changeable design decisions because these design decisions transcend. Execution time, design modules may not correspond to processing steps in the implementation of the system.

Information Hiding is of much significance when modifications are required during the testing and maintenance phase. In object oriented design it gives rise to the concepts of encapsulation, modularity and is also linked to the concept of abstraction. Information hiding also hides the following information’s:

- (i) A data structure, its internal linkage and the implementation details of the procedure.
- (ii) The format of control blocks such as those for queues in an operating system.
- (iii) Character codes, ordering of character sets, shifting masking and other machine dependent details.

Advantages of Information Hiding

- (i) It reduces the likelihood of adverse effects.
- (ii) Leads to low coupling.
- (iii) Attaches importance to communication through controlled interfaces.

- (iv) Limits the global impact of local design decisions.
- (v) Helps in producing higher quality software.

5.7 DESIGN QUALITY METRICS

Design quality metrics is a methodology of determining the quality of a design. A design is supposed to be good if it implements a specification correctly and allows efficient code to be produced. A good design is one which is maintainable and understandable. A maintainable design can be adapted to modify existing functions and add new functionality. The design components should be cohesive as they should have close logical relationship. They should be loosely coupled and should not be tightly integrated. Design quality metrics are used to assess the quality of design. Coupling & cohesion measures have been developed in conjunction with functional.

5.7.1 Cohesion

The internal cohesion of a module is measured in terms of the close of the relationship between its components or in other words as the strength of binding elements within the module. Cohesion occurs on the scale of weakest elements (least desirable) to strongest elements (most desirable). It shows how tightly the elements of the module are attached to one another. A component should implement a single logical function or a single logical entity. Contribution to implementation should be from parts of components. There is no need to modify many components if a change is desired. Cohesion gives the designer an idea about whether or not the different elements of a module belong together in the same module.

Constantine and Yourdon identify seven different levels of cohesion in order of increasing strength:

1. Coincidental	2. Logical
3. Temporal	4. Procedural
5. Communicational	6. Sequential
7. Functional	

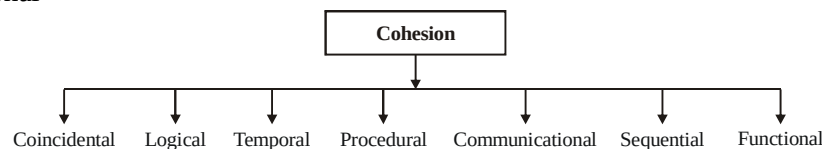


Fig. 5.7: Cohesion

1. **Coincidental:** If occurs when there is no meaningful relationship among the elements of a Module. Unrelated functions, processes or Data are present in the same component for the reasons of easiness serendipity. Coincidental cohesion can happen if an existing program is 'modularized' by cutting it into pieces and creating different pie modules. If a module is created to save duplicate code by combination some part of the code that occurs at many different places that module may have coincidental cohesion. In General, it is not wise to create a most merely to avoid duplicate code or chop a module to reduce the module size.

2. **Logical Cohesion:** A module is said to have logical cohesion if there is logical relationship between the elements of a module. Several logically related functions or data elements are kept in the same component. A typical example of this type of cohesion is a module that performs all the inputs or all the outputs. One component may read all types of input (from tape, disc or telecommunications port). It does not matter where the input is coming from or how it will be used. Although it is more reasonable than coincidental cohesion, the elements of a logically cohesive component are not related functionally. In general, logically cohesive modules should be avoided, if possible.

3. **Temporal Cohesion:** Temporal cohesion and logical cohesion are similar except that in temporal cohesion the elements a related in time and are executed together. Modules that perform activities like "Initialization" "clean up" and "Termination" a generally temporally bound. This avoids the problem of passing the flag and the code is generally simpler.

4. **Procedural Cohesion:** A procedurally cohesive module contains elements that belong to a common procedural unit. Sometimes it is needed that functions must be performed in a certain order e.g.

Data must be entered before they can be checked and then manipulated; all these three functions in a specific order. When functions are grouped together in a component to maintain a proper procedure, the component is procedurally cohesive. Such modules occur when modular structure is determined from same form of flow chart. Procedural cohesion, often, cuts across functional times.

5. **Communicational Cohesion:** In this type of cohesion, elements are related to each other by a reference to same input or output data because they operate on or produce the same data set. Sometime, unrelated data are fetched together because it can be done with only one disc or tape access. Communicably cohesive modules can do more than one function. e.g. “Print and Punch record”. It is to be noted that sometimes communicational cohesion may destroy the modularity and functional independence of the design.

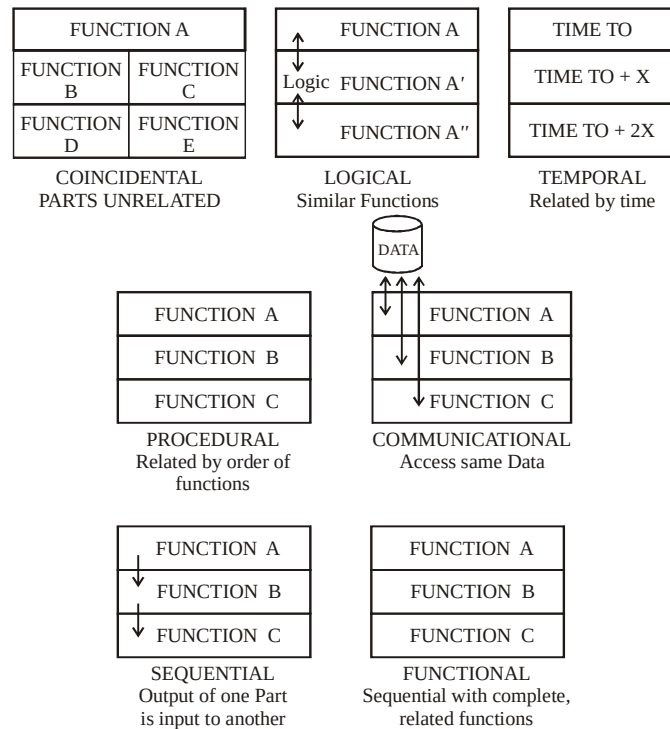


Fig. 5.8: Examples of Cohesion

6. **Sequential Cohesion:** When the output from one part of a component forms the input to the next part, we get sequential cohesion. Sequential cohesion does not provide any directions on how to combine them into modules. Different possibilities exist:

- Combine all in one module.
- Put the first half in one and the second half in another.
- The first third in one and the rest in the other and so on sequentially cohesive modules bear a close resemblance to the problem structure however they are still considered to be far from the ideal; which is function cohesion.

7. **Functional Cohesion:** Functional cohesion is the strongest cohesion. In this type of cohesion, every processing element is necessary for the performance of a single function and all the desired elements are contained in one component. A functionally cohesive component performs only and only that function for which it is designed and nothing else.

A cohesive object is one in which a single entity is represented and all the operations on that entity are included with the object. Such a cohesion called object cohesion is a new class of cohesion in which each operation provides functionality which permits the characteristics of the object to be modified, inspected or used as a basis for service provision.

5.7.2 Coupling

Coupling is the degree to which each program module relies on each one of the other modules. Coupling is a property of a collection of modules. It is an indication of the strength of the inter connections between the components in a design. Highly coupled systems have strong inter-connections with program units dependent on each other. Loosely coupled systems on the other hand are made up of components which are independent or almost independent. Loose coupling is desirable for good software engineering but tight coupling may be necessary for maximum performance. Uncoupled components have no inter connections at all coupling depends on several things:

- (i) The references made from one component to another.
- (ii) The amount of data passed from one component to another.
- (iii) The amount of control one component has over the other.
- (iv) The degree of complexity in the interface between components.

The following diagram showing component coupling explains everything. Dependency between modules has been shown by taking four modules A, B, C and D.

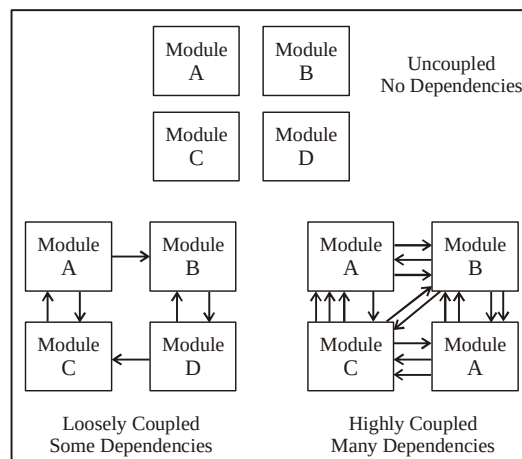


Fig. 5.9: Coupling

Coupling can be measured from complete dependence to complete independence. Actually a system can not be built of completely uncoupled components. When coupling is high, large parts of the system may be perturbed by the change. Low coupling helps in minimizing the number of components requiring revision.

5.7.2.1 Types of Coupling

Some types of coupling, in order of highest to lowest coupling are as follows: 1. Content coupling, 2. Common coupling, 3. External coupling, 4. control coupling, 5. Stamp coupling, 6. Data coupling message coupling

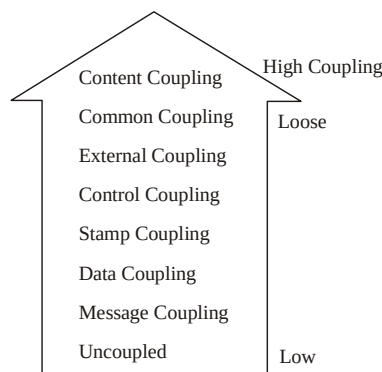


Fig. 5.10: The Range of Coupling Measures

1. **Data Coupling:** Two modules are data coupled, if they communicate by using an elementary data item that is passed as a parameter between the two, e.g. an integer, a float, a character etc. Data coupling is simpler and leaves less room for error. It is most convenient to trace data and to make changes. If coupling must exist between components data coupling is the most desirable.
2. **Message Coupling:** This is the loosest type of coupling. Modules do not depend on each other; instead they use a public interface to exchange parameter less messages.
3. **Stamp Coupling:** Stamp coupling is when modules share a composite data structure and use only a part of it, possibly a different part (e.g. whole record is passed to a function which needs one field of it).
4. **Control Coupling:** Two modules are control coupled if they pass a piece of information intended to control the internal logic using at least one 'control flag'. The control flag is a variable that controls decisions in subordinate or superior modules. The advantage of control coupling is that each component controls only one function or executes one process. This restriction is necessary to minimize the amount of controlling information which must be passed from one component to another and localizes control to a fixed and identifiable set of parameters forming a well-defined interface.
5. **External Coupling:** External coupling occurs when two modules share an externally forced data format, communication protocol or device interface.
6. **Common Coupling:** Two modules are common coupled if they both are the same global data area. Changing the shared resource implies changing all the modules using it. The amount of coupling can be reduced by organizing the design so that data are accessible from a common data store but dependence is still there because making a change to the common data means tracing back to all components that access the data to calculate the effect of that change. This kind of coupling is known as common coupling. With common coupling it becomes difficult to find which component is responsible for having set a variable to a particular value.

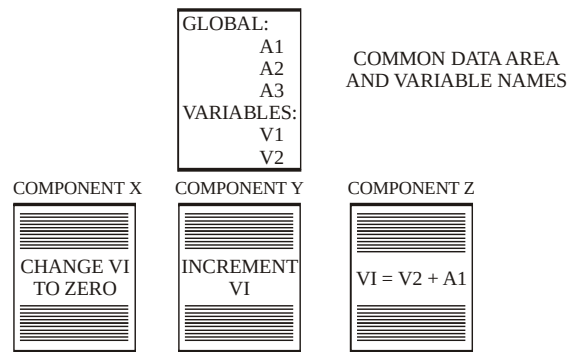


Fig. 5.11: Example of Common Coupling

7. **Content Coupling:** Content coupling exists between two modules if these module changes a statement in another module, one module references or changes data contained inside another module, or one module branches into another module. One module actually modifies the procedural contents of another module. In content coupling, modules are inter dependent on each other.

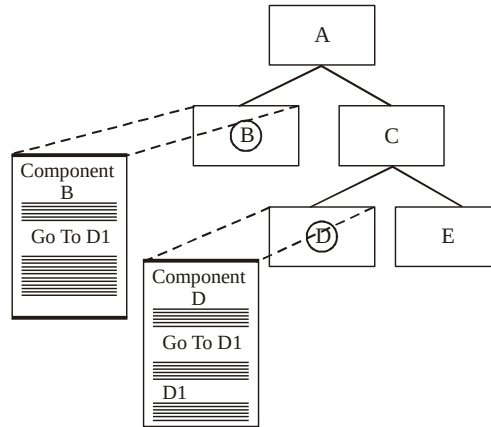


Fig. 5.12: Example of Content Coupling

5.8 DATA DESIGN

Data design is the first of all the activities in a software design process. It leads to a better program structure, effective modularity and decreased complexity. Data Design is developed by the transformation of the Data dictionary and entity relationship diagram into Data Structure which are necessary to implement the software. The whole process includes identifying the data defining specific data types and storage mechanisms and guarantees data integrity by using business rules and other mechanisms.

Data design creates a model of data information that is represented at a high level of abstraction i.e. the customer or user's views of data. The data model is refined into progressively more implementation specific representations which can be easily processed by the computer based system. In many software applications, the architecture of the data will put great influence on the architecture of the software that must process it.

The selection process may take the help of algorithmic analysis of alternative structures in order to determine the most efficient design or it can involve the use of a set of modules that provides operations on some representation of objects. Some principles are observed while specifying the data are given below:

1. The Systematic analysis principle applied to function and behavior should also be applied to data. Lot of time and energy is spent on deriving, reviewing and specifying functional requirements and preliminary design. In addition to it representations of data flow and content should be developed and reviewed; data objects should be identified, alternative data organizations should be considered and the impact of data modeling on software design should be evaluated.
2. All data structures and the operation to be performed on each should be identified. The design of an efficient data structure depends on operations to be performed on data structure upon evaluation of the operations; an abstract data type is defined for use in subsequent software design which simplifies software design considerably.
3. A data dictionary should be established and used to define both data and program design. The specific relationship among data objects and the constraints on the elements of a data structure can be more easily defined by dictionary like data specifications.
4. Low level data design decisions should be deferred until late in the design process. Overall data organization can be defined during requirements analysis, refined during data design work and specified fully during component level design.
5. The representation of the data structure should be known only to those modules that must make direct use of the data contained within the structure. The quality of a software design is studied by the concept of information hiding and concept of coupling.
6. A library of useful data structures and the operations that may be applied to them should be developed which can reduce both specification and design effort for data.

7. A software design and programming language should support the specification and realization of abstract data types.

There are three levels to view the structure of data:

1. Program component level
2. Application Level
3. Business Level

At the program component level, design of data structures and the algorithms needed to manipulate them are essential for the development of a high quality design.

At the application level, the translation of a data model into a data base is necessary to get the specified business objectives. At the business level, the collection of information stored in different data bases should be reorganized into data ware house which enables data mining that has an effect on the business.

5.9 ARCHITECTURAL DESIGN

An architectural design acts as a preliminary “blue print” from which software is developed. Architecture describes the software top-level structure which identifies its components. IEEE architectural design as ‘the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.’

First of all software requirement is examined to establish a framework and then building a physical model using recognized software engineering methods. Shaw and Garlan (1996) suggest that software architecture is also the first step in producing a software design levels; architecture, code design and executable design.

It is very useful to work from the top-down designing an architecture, then the code design and finally the executable design but Krasner, Curtis and Iscoe (1987) have studied the habit of developers on 19 projects; they present a report and on the basis of other evidences confirm that actually the designers more back and forth from one level to other as they get extra knowledge about the solution and its implication. As the designers explore other aspects of the system, the designers may interact with testers or programmers bringing changes in the design to step up implementation, testability or maintainability.

In Architectural design performs the following functions:

1. The software designers can specify the system behavior (function and performance) after the architectural design provides a level of abstraction.
2. It is really the consciences for a system, a good architectural design guides the process of system enhancement indicating what changes can be made without compromising the integrity of the system.
3. It evaluates all top level designs.
4. It develops and documents top level design for the external and internal interfaces.
5. It develops preliminary versions of user documentation.
6. It defines and documents preliminary test requirements and the schedule for software integration.

5.10 PROCEDURAL DESIGN (FUNCTION ORIENTED DESIGN)

Procedural design techniques are very common and popular and at present they are being used in many software organizations. It is the result of focusing full orientation to the function of the program. This is an approach to software design where the design is decomposed into a set of interacting units where each unit of module performs a specific function Niklans Wirth who is the creator of Pascal and a number of other languages are one of the best known advocates of this method. His special variety to known as step-wise refinement. This stepwise refinement is a top-down strategy where a program is refined as a hierarchy of increasing levels of detail (Mills 1988). Refinement is actually a process of elaboration.

Procedural design is also named as component level design. It occurs after data, architectural and inter face designs have been developed. The aim is to translate the design model into an operational software. The translation from high level of abstraction of existing design into low level of abstraction of

operational program is risky because it can bring in subtle errors that are difficult to locate and correct in later stages of the software process. During this translation, certain design principles must be followed. Regardless of the methodology adopted to represent the procedural design, the data structures, interfaces and algorithms defined should follow to a variety of well-established procedural guiding principles to avoid errors as the procedural design is developed.

1. **Structured Programming:** In the late 1960s, Dijkstra and others gave the idea of using a set of constructs from which any program could be formed. Each construct had a predictable logical structure which was entered at the top and exited at the bottom for the convenience of the reader to follow procedural flow more easily. These constructs are Sequence, Condition and Repetition. The sequence implements processing steps, condition provides facilities for selected processing and Repetition allows for looping. These structured constructs are meant to limit the Procedural Design to a small number of pre-determined operations this contributes more to a human understanding of the process called chunking by the psychologists. Any program needs three constructs only for designing and implementation.

5.11 STRUCTURED DESIGN METHODOLOGY (SDM)

Structured Design Methodology helps the designer during process. Software is a transformation function that converts given inputs to desired outputs. The main objective of this strategy is to specify functional modules and connections. SDM controls and puts an influence on the final design so that the programs implementing design would have a nice hierarchical structure that has functionally cohesive modules and few inter connections between modules as far as possible. SDM aims to achieve high cohesion but low coupling.

Steps: SDM follows the following procedure:

1. **Restate the problem as Data Flow Diagram (DFD):** The Designer finds full freedom in creating a DFD because the system does not exist yet. DFD will solve the problems stated in SRS.

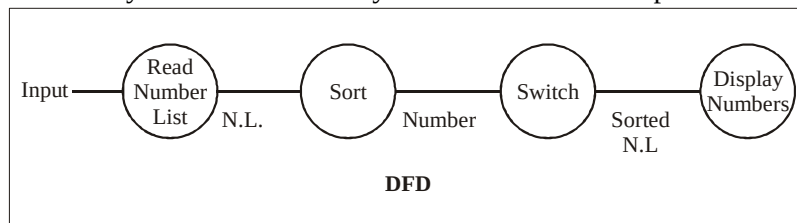


Fig. 5.13: Identification of DFD

Let us take the example of simple sort program that sorts the 'n' numbers in ascending order. The problem has only one input file i.e. Read Number List. If we want to transform the input to the desired output, all the numbers are stored in a list. Then all numbers list goes to the sort transformation. If numbers are not sorted, Switch Transform changes the place order of number. Sorted Number are then passed to display transformation for printing numbers in ascending order on the screen.

2. **Identify most Abstract Input (MAI) and Most Abstract Out (MAO) data elements:** While going from outputs to inputs, the output of the major transformation is known as MAO and the input of major transformation is known as MAI. In this example of sort program, Number is MAI to switch transform and sorted N.L. is MAO.

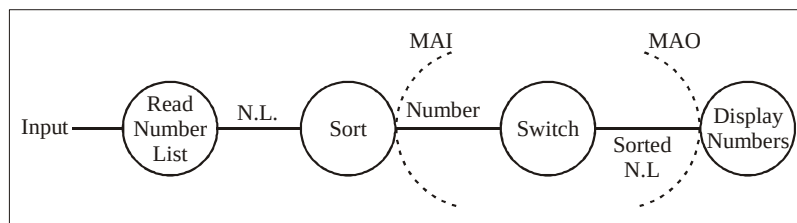
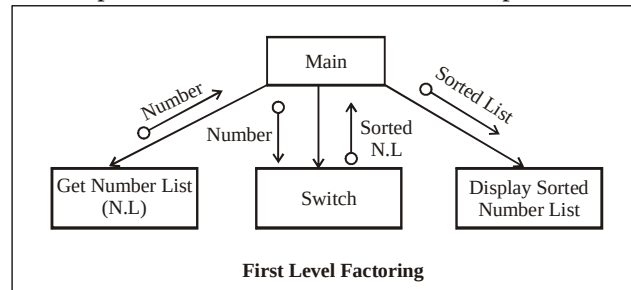


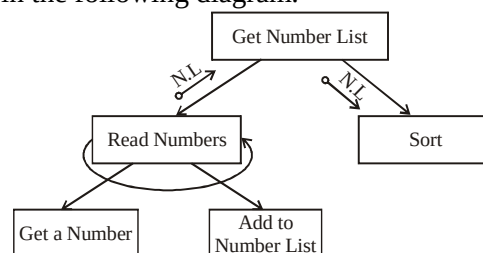
Fig. 5.14: Identification of MAI & MAO

3. **First Level Factoring:** After the identification of central transformation MAI and MAO; the next step is the analysis of these information's. Main Module is specified first of all whose aim is to invoke the subordinates. For each MAI data item, an immediate subordinate module to the main module is specified. Each module is an input module which aims to provide to the main module the most abstract data item for which it is created. Same procedure is adopted for MAO data item, a subordinate module that is an output module that accepts data from the main module is specified.



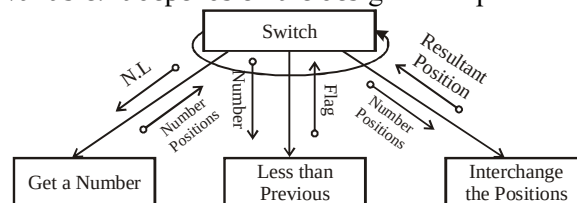
4. **Factoring the input, output and transform branches:** Each module in DFD is factored to move details:

(a) **Factoring the input Module:** The objective of input module is to produce some data to the main module. Do factor as input module, the transform in the DFD that produces number list is treated as a central transform as is shown in the following diagram.



(b) **Factoring the output Module:** It resembles the factoring of input module. The aim is to reach the final output. During the factoring of output module, usually there will be no input module because this task is performed by MAI.

(c) **Factoring the Central Module:** To factor a central module into its subordinate modules; there are no standard guidelines available. It depends on the designer's experience:



Graphical Design Notation

In this a flow chart is quite simple in the form of pictures.

A box indicates processing steps, a diamond represents a logical condition and arrows show the flow of control. Condition (if then else) is represented by decision diamond – if true – it causes 'then part processing' and if false – it invokes 'else part processing'.

Repetition is depicted by two slightly different forms. The do-while tests the condition and executes a loop task again and again as long as the condition holds true. A repeat until executes the loop task first and then tests a condition and performs the task again until the condition fails. The selection construct is the extension if then else successive decisions are taken to test a parameter until a true condition occurs and a case part processing path is executed. The structured constructs may be nested within one another. By nesting constructs in this way, a complex logical scheme may be developed.

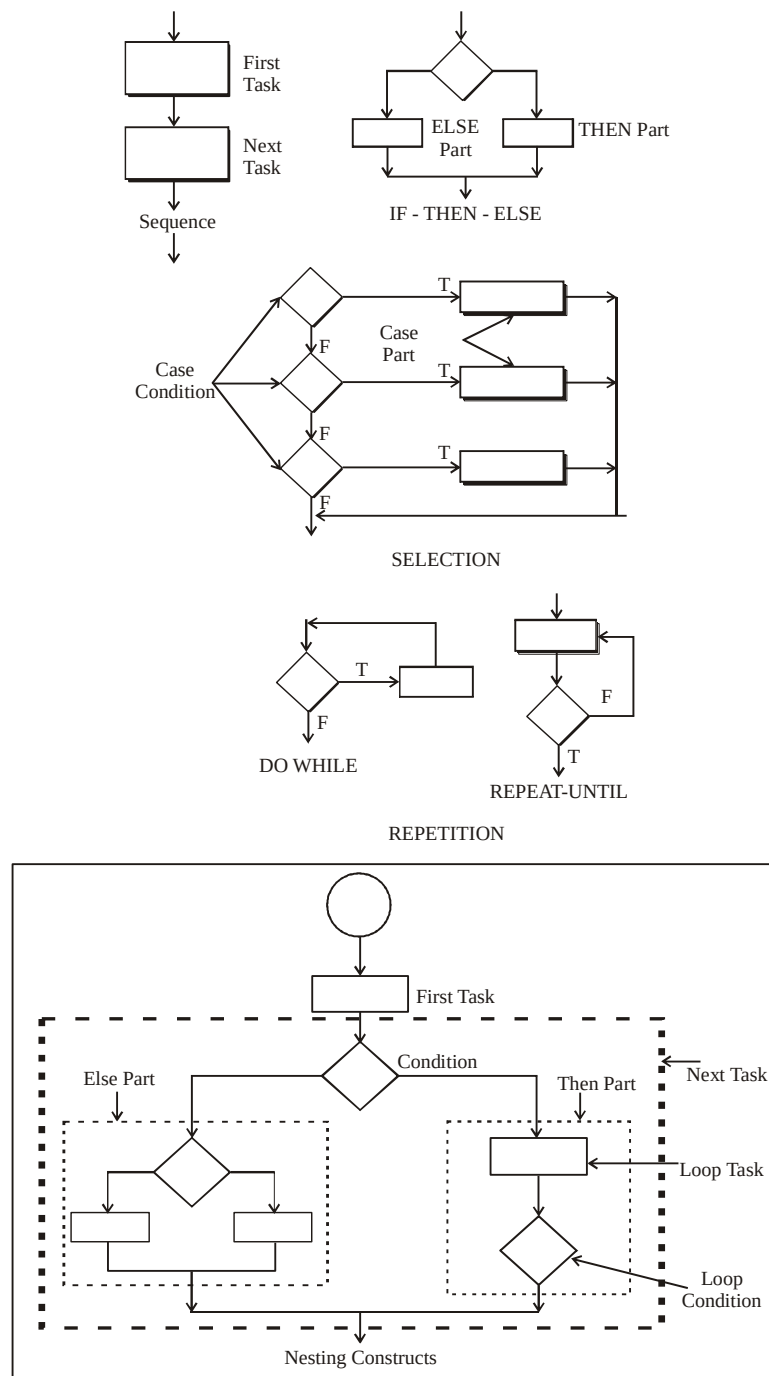


Fig. 5.15: Nesting of Different Structured Constructs

Tabular Design Notation: A module may be required to evaluate a complex combination of conditions and select proper actions based on these conditions. Decision tables provide criteria that translate actions and conditions in a tabular form. The table is not easy to misinterpret and if is possible to be used as a machine readable input to a table driven algorithm. Decision table is divided into four sections. As is clear from the following diagram; the upper left quadrant contains list of all conditions and

the lower left hand quadrant contains a list of all actions that are possible depending on combinations of conditions. The right hand quadrants indicate condition combinations and the related actions that will happen for a specific combination.

CONDITIONS	1	2	3	4					n
Conditions # 1	✓			✓	✓				
Conditions # 2		✓		✓					
Conditions # 3			✓		✓				
ACTIONS									
Action # 1	✓			✓	✓				
Action # 2		✓		✓					
Action # 3			✓	✓					
Action # 4			✓	✓	✓				

Fig. 5.16: Decision Table Nomenclature

3. **Program Design Language (PDL):** PDL is also known as structured English or pseudo code. This is a language that uses the vocabulary of a natural language (English) and the overall syntax of a programming language. IEE defines PDL as a specification language with special const and sometimes with special constructs and sometimes verification protocols used to develop, analyze and document a program design. A team of designers and programmers use PDL as a way to ensure that actual programming is likely to match design requirements PDL is mainly used during the detailed design phase and uses to describe algorithms and specify interfaces between modules P&L enables the programmer to concentrate on the algorithms without worrying about all the syntactic details of a particular programming language.

Advantages of using PDL

- It uses English like statements that describe specific operator very precisely.
- It supports the idea of iterative refinement.
- It is expressed in easy-to-read format.
- Reviews became very easy because the source code is no examined.
- The language uses flow chart replacements, program documentation and technical communication at all level.
- Continuous use and refinement of the PDL has established if as the medium of choice for either creating refining a detailed program.

5.12 DESIGN VERIFICATION

Software design verification or review is necessary to review or check the design, implementation way, testing and maintenance, members of Development team. These reviews are well documented, comprehensive and systematic evaluation of a design to ensure the adequacy of the design requirements. It is done to evaluate the capability of the design and to identity problems IEEE defines if as a formal meeting at which a system's preliminary or detailed design is presented to the user customer or other parties for comment and approval.

The main aim behind the design review is to detect the errors in the design process which will reflect in code and also in the final system. These verifications are done to check if the design is of good quality or not. These reviews are taken at the end of the design phase to find solutions and resolve issues regarding software related design decisions, architectural design and detailed design. The reviews include study of the development plans, requirements and design specifications, testing plans and procedures, verification results for each stage of the design.

Types of Reviews: The review step generally involves three steps:

1. Preliminary Design Review
2. Critical Design Review

3. Program Design Review

1. **Preliminary Design Review:** It is a formal inspection of the high level architectural design to satisfy whether the design satisfies the functional as well as non-functional requirements of the customers and users. The review team includes customers, moderator, secretary, system designers, other state holders not involved in the project. The main aim is to:

- (i) To specify whether effective modularity is achieved or not.
- (ii) To define interfaces for modules and external system elements.
- (iii) To ensure maintain ability has been considered or not.
- (iv) To assess quality factors.
- (v) To check discrepancies if any and to resolve it by the review team.

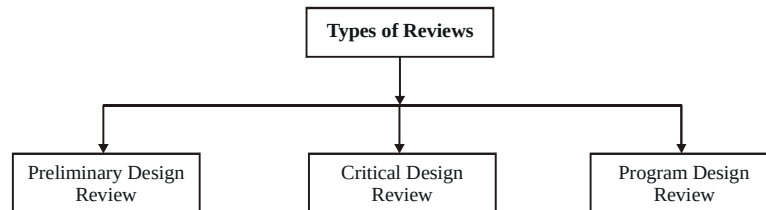


Fig. 5.17: Types of Reviews

2. **Critical Design Review:** Critical Design Review is taken after the successful completion of the preliminary design review. After the customer is satisfied, a critical design review is conducted with the following purpose in mind:

- (i) To ensure that designs (conceptual and technical) are without defects.
- (ii) To determine whether the design satisfies the requirements specifications or not.
- (iii) To assess the functionality and maturity of the final design.
- (iv) To justify that the design has clarity, effectiveness and easy to understand. A review is formed to check the design specifications using diagrams and data. Other individuals in addition to the members of the review team are:
 - (a) System tester: A technical expert.
 - (b) Analyst: Responsible for writing system documentation.
 - (c) Program Designer: Who understands the design in detail.

If a minor fault is found, it is resolved by the review team. In case of a major fault, the review team may take decision to revise the proposed technical design.

3. **Program Design Review:** The designers and the developers conduct this review after the completion of Critical Design Review. This is done to get feedback on the designs before implementation begins. The main purpose is to:

- (i) Ensure the feasibility of the design.
- (ii) Ensure that the interface is consistent with the architectural design.
- (iii) Specify whether the design is amenable to implementation language.
- (iv) Ensure that structured programming constructs are used throughout Software Design Process.
- (v) Ensure that members of the implementation are able to understand the proposed design.

The team of Program Design Review includes program designers, developers, a system tester, moderator, secretary and analyst. A successful Program Design Review presents all plans related to coding plans before coding begins.

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 6: CODING

6.1 INTRODUCTION

6.2 CODING

6.3 CODING STANDARDS

6.4 CODING GUIDELINES

6.5 PRINCIPLES OF PROGRAMMING/CODING

6.6 CODE REVIEW/VERIFICATION

6.1 INTRODUCTION

The software coding deals with design of data structure, information hiding, implementation. It is that phase of software development where actual implementation is carried out. The coding is done in a team manner which requires standardization of coding style, coding guidelines and coding documentations. The information hiding is another important aspect which is handled in this phase. In this chapter, we will go through the different coding styles and guidelines.

6.2 CODING

1. After the completion of design phase, coding is undertaken. The objective of the coding phase is to transform the design of a system into high level language code and then to perform unit testing at this code.
2. For adequate and effective coding, programmers have to mandatorily follow the coding standards. Before the testing phase, coding standards should be verified.
3. Coding guidelines provide some general suggestions regarding the coding style to be followed.
4. After the coding of a module, code reviews are carried out. This code review ensures that coding standards are followed during coding of a module. This code review also detects as many errors possible before testing.
5. It is important that the code should be written in such a manner that it could be modified later, if the need be.

6.3 CODING STANDARDS

A coding standard is a list of several rules to be followed during coding like variables should be named, name of module, exception handling mechanism etc. Many organizations develop their own coding standards and guidelines according to their need and type of software product they are going to develop. The some commonly known coding standards are as follows:

1. The organizations should follow the exact format of the header information which is as follows:
 - Module's Name
 - Author's Name
 - Module's Creation Date
 - Module's Synopsis
 - Input/output parameters with different functions

2. There are some rules which limits the use of global. These rules explains what type of data can be declared global and what type of data cannot declared global.
3. The method of reporting error conditions and exception conditions should be standardized within an organization.
4. Global variable's name should always start with a capital letter whereas local variable's name should always start with small letter.

6.4 CODING GUIDELINES

Coding guidelines provides some general suggestions regarding the coding style to be followed. Some commonly known coding guidelines are as follows:

1. The function should be small not lengthy because length function is difficult to understand and arises large number of bugs.
2. goto statements should be avoided because it makes program unstructured.
3. The code should be well documented. It is only possible by using comment lines which explain it appropriately.
4. The code should be easy to understand. The complex coding is difficult to understand and maintain.
5. Avoid to use same identifies for multiple purposes. Same identifier for multiple purposes makes future enhancements very difficult and lead to confusion.
6. Variable names should be clear and simple. Variable names should be mnemonic procedure and function name should also be mnemonic.

6.5 PRINCIPLES OF PROGRAMMING/CODING

Programming is the process of writing, testing, debugging and maintaining of a computer program. The following are some basic principles of good programming practice.

1. Before start the programming, developer should understand the problem. He should choose a programming language which meets the needs of the software to be built and the environment in which it will operate.
2. Developer should create a visual layout that aids code understanding.
3. Avoid repetition in programming.
4. Coding should be simple because it takes less time to write, easier to modify and has fewer bugs.
5. Coding should improves code reliability and decrease development time
6. Developer should create a set of unit tests that will be applied once the coding is complete. Perform unit tests and correct the uncovered errors.
7. Developer should conduct a code walkthrough and refactor the code.
8. Developer should write the code for maintainer because any code that is worth writing is worth maintaining in the future, either by him(developer) or by someone else.

6.6 CODE REVIEW/VERIFICATION

1. After the coding of a module, code reviews are carried out. This code review ensures that the coding standards are followed during coding of a module. This code review also detects as many errors possible before testing.
2. After the successful compilation of module, code review takes place. Code review is a cost effective strategy which produce high quality code by eliminating coding errors.
3. Eliminating an error from code involves three main activities—testing, debugging and then correcting errors. Debugging is very laborious and time consuming activity than testing.

4. There are four types of code review which are as follows:

- (i) Code Walk through
- (ii) Code Inspection
- (iii) Clean Room Testing
- (iv) Static Analysis

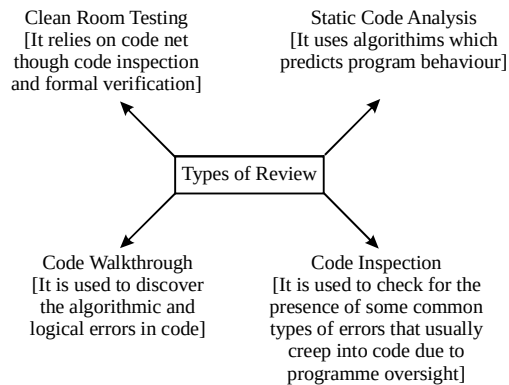


Fig. 6.1 Types of Code Review/Verification

1. **Code Walkthrough**

- Code walkthrough is an informal code analysis technique.
- Its aim is to discover the logical and algorithmic errors in the code.
- It does not tell how to fix the discovered errors.
- Before the code walkthrough meeting, the members note down their findings to discuss.
- The success of code walkthrough are based on personal experience, common sense and several subjective factors.
- The walkthrough team should consist of three to seven members. The walkthrough team should not be too big or too small.

2. **Code Inspection**

- Code inspection is a formal code analysis technique.
- The aim of the code inspection is to discover common types of errors which are caused due to improper programming.
- Code inspection detects the commonly made errors along with the coding standards.
- It is also used to check whether coding standards are followed during the coding or not.
- Code inspection checks some programming errors like incompatible assignments, improper storage allocation and de-allocation, improper modification of loop variables, non-terminating loops etc.

3. **Clean Room Testing**

- It relies on code walkthrough code inspection and found verification.
- The aim of clean room testing is to produce documentation and code which is more maintainable and reliable.
- Clean room testing involves walkthroughs, inspection and formal verification which is time consuming and hence testing effort is increased.

4. **Static Analysis/Static Code Analysis**

- It is also known as automated code review.
- Static code analysis is Excellent for enforcing compliance to standards
- It Identifies bugs in code and the design and implementation problems
- It uses algorithms which predict program behaviour.

- Static analysis explain the purpose of possible executions of a program. It gives assurance about any execution but this tool spend a lot of effort dealing with developer.
- Static analysis can be performed on modules or unfinished code

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 7: SOFTWARE TESTING

7.1 INTRODUCTION

7.2 SOFTWARE TESTING

7.2.1 OBJECTIVES OF SOFTWARE TESTING

7.2.2 PRINCIPLES OF SOFTWARE TESTING

7.3 BASIC CONCEPTS

7.4 SOFTWARE DEBUGGING

7.5 PLANNING OF TEST

10.6 SOFTWARE VERIFICATION AND VALIDATION

7.7 BLACK BOX TESTING

7.8 WHITE BOX TESTING

7.9 GREY BOX TESTING

7.10 DIFFERENCE BETWEEN BLACK BOX TESTING, GREY BOX TESTING AND WHITE BOX TESTING

7.11 LEVELS OF TESTING

7.11.1 UNIT TESTING

7.11.2 INTEGRATION TESTING

7.11.3 SYSTEM TESTING

7.11.4 ACCEPTANCE TESTING

7.1 INTRODUCTION

Software testing is an activity that has significant impact on the software development process. It is carefully designed process and must be performed effectively. Since allocated resource is limited and time is another constraint, so it is not possible to test any system with all possible inputs. It is the primary quality-control measure used during software development. The testing provides methods and techniques to test a system in reasonable time with assurance to give quality performance. The testing is broadly categorized as Black Box testing and White Testing. There is another way to categorize the testing method called level testing discussed in detail in this chapter. The chapter includes detail discussion on testing along with advantages and limitations.

7.2 SOFTWARE TESTING

1. Testing is an important and critical activity of software development life cycle. It is the primary quality control measurement activity used during software development.
2. Testing is a set of systematic activities that can be planned in advance and conducted with the intent of finding errors. It makes us feel confident that the software works well.
3. The goal of testing is to uncover requirement, design, and coding errors in the program. It is critical, expensive and time-consuming activity.

4. The testing process requires proper planning. The testing cannot show the absence of defects. It can show only software errors present.
5. During the testing phase, emphasis should be on the following:
 - Tests should be planned long before testing begins.
 - All tests should be traceable to customer requirements.
 - Tracing should begin “in the small” and progress toward testing “in the large.”
 - Testing must be conducted by the independent third parties.

7.2.1 Objectives of Software Testing

The testing objective is to test the code with aim to discover errors. It also expresses that the software functions are working according to software requirements specification with regard to functionality, features, facilities and performance. It is here noticeable, that testing will detect errors in the written code but it will not give you an idea about an error if the code does not address a specific requirement set in the SRS.

The primary objectives of testing are:

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding the undiscovered error.
3. A successful test is one that uncovers an as-yet-undiscovered error.
4. It makes us feel confident that the software works well.

7.2.2 Principles of Software Testing

The testing principle is a guide book which helps software engineer to design effective test cases. There are many principles that a software engineer must understand before starting the testing process.

The following are the main principles of testing:

Sr. No.	Principles	Description
1	All tests should be traceable to customer requirements	It states that any defects that might cause the program or system to fail to meet the client's requirements must be uncovered.
2	The Pareto principle applies to software testing.	It is also known as 80/20 principle. It states that 20% of the problems lead to 80% of other problems. One should concentrate to 20% of the problems and 80% of the problems will automatically remove.
3	Tests should be planned long before testing begins.	According to principle testing phase should be started soon after the requirements model is completed. The detailed test cases can begin as soon as the design model is designed.
4	Testing should begin “in the small” and progress toward testing “in the large.”	It states that the first test may be focused on individual components. As testing progresses, focus shifts in an attempt to find errors in integrated components and later on full system.
5	Testing should be conducted by an independent third party	The principle states that the testing should be conducted by an independent third party. The software engineer who has developed the system is not the best person to conduct testing
6	Exhaustive testing is not possible.	It states that It is possible, however, to adequately cover program logic and to ensure that all conditions in the component-level design have been exercised for testing.

Table 7.1 Principles of Software Testing

7.3 BASIC CONCEPTS

In software testing, there are few terms which generally create confusion. These terms are explained as follows:

1. **Errors:** An error is a discrepancy between the actual value of the output given by the software and the specified (correct) value of the output for that given input.

Errors occur when any aspect of software product is incomplete or incorrect. It is a mistake committed by the development team during development process. Errors can be classified into following categories:

- **Implementation Errors:** These errors made in translating design specifications into source code.
- **Requirement Errors:** These errors are due to incomplete and incorrect statements of user requirements. User fails to specify functional and nonfunctional requirements.
- **Design Errors:** These are introduced by failure to translate the requirements into complete and correct structures.
- **Syntax Error:** A syntax error is a program statement that violates one or more rules of the language in which it is written.
- **Logic Error:** A logic error deals with incorrect data fields, out-of-range terms and invalid combinations.

2. **Fault:** A fault is a condition that causes a system to fail in performing its required function. A fault is the basic reason for software malfunction.

It is also commonly called a bug. Sometime, it is possible that even providing correct input system give wrong results which due to some bug in the system. When it fails then we say that the system has a fault or a bug and it needs repair.

3. **Failure:** Failure is the inability of the software to perform a required function to its specification.

In other words, when software goes ahead in processing without showing error or fault even though certain input and process specification are violated, then it is called a software failure.

4. **Quality Assurance:** Quality assurance is the planned and systematic approach. It ensures that the standards, processes and methods are established and followed throughout the software development. Quality assurance is a set of activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.

5. **Quality Control:** Quality control is used to verify the quality of the output. Quality control is set of activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.

6. **Testing:** Testing is the sequence of activities which ensure the identification of bugs in the software.

Difference between Testing, Quality Assurance and Quality Control

Quality Assurance	Quality Control	Testing
Quality assurance focuses on processes and procedures rather than conducting actual testing on the system.	Quality control focuses on actual testing by executing software with intent to identify bug/defect through implementation of procedures and process.	Testing focuses on actual testing of the system or unit with suitable test
Quality assurance is process oriented activities.	Quality control is a product oriented activities.	Testing is product oriented activities.
Quality assurance preventive activities.	Quality control is a corrective process.	Testing is a preventive process
It is a subset of Software Test Life Cycle (STLC)	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

Table 7.2 Difference between Testing, Quality Assurance and Quality Control

7. **Test Oracles:** The test oracle defines a method to understand the determined and expected behavior of a system in order to test the test case for checking abnormality in behavior. In other words, a

test oracle is a mechanism, different from the program itself, which can be used to check the correctness of the output of the program for the test cases.

The following diagram shows how test oracle is useful to determine the expected behavior of the system

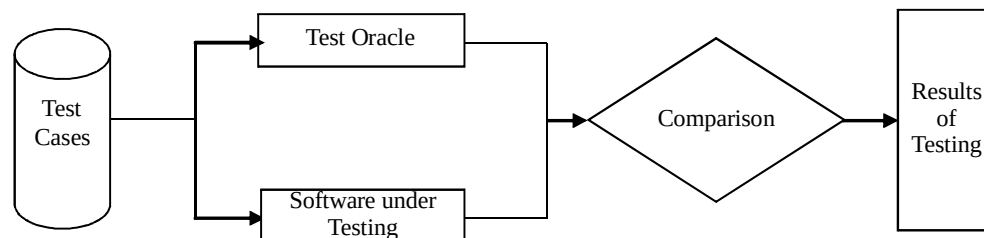


Fig 7.1 Working of Test Oracle

The above diagram shows testing process in which the test cases are given to the test oracle and the program under testing. The output of the two is compared to determine if the program behaved correctly for the test cases.

The human oracles generally use the specifications of the program to decide what the correct behavior of the system should be. To help the oracle to determine the correct behavior, it is important that the behavior of the system be unambiguously specified and the specification itself should be error free. Test oracles are human beings, so they may make mistakes when there is a discrepancy between the oracles and the results of a program. We need to verify the result produced by the oracle before declaring that there is a fault in the program.

8. **Software Testability:**

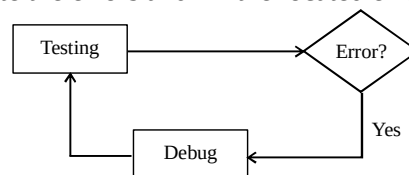
- Software testability is how easily a computer program can be tested.
- It is used to measure how adequately a particular set of tests will cover the product.
- The characteristics which leads to testable software are: operability, observability, decomposability, simplicity, stability and understandability.

9. **Attributes of a Good Test:** The following are some attributes of good test:

- A good test has higher chances of finding errors.
- A good test is neither too simple nor too complex.
- A good test takes less time and resources.
- A good test is not redundant.

7.4 SOFTWARE DEBUGGING

Testing and debugging are different terms. Debugging is the process of locating and fixing errors in program code. Testing only determine the errors in the code. Testing does not fix those identified code. Whereas debugging is used to locate the errors and fix the located errors.



Difference between Testing and Debugging

The testing involves the identification of bugs in the program without correcting it. Normally people with a quality assurance background are involved in the identification of bugs. Testing is performed in the testing phase. The debugging involves fixing the bugs. The developers who coded the program conduct debugging.

Testing	Debugging
Testing is a process of finding and locating of a defect in a program	The debugging is process of fixing the defects in the program
Testing is done by the testing team	The debugging is done by the development team
Testing is done with intention to find as many defect as possible	The debugging is focused in removing the defects

Table 7.3 Difference between Testing and Debugging

7.5 PLANNING OF TEST

A test plan is a document consisting of different test cases designed for different testing objects and different testing attributes. The plan puts the tests in logical and sequential order per strategy chosen, top-down or bottom-up. The test plan is a matrix of test and test cases listed in order of its execution.

A test plan states:

- The items to be tested.
- At what level they will be tested at.
- The sequence they are to be tested in.
- How the test strategy will be applied to the testing of each item and the test environment.
- The software tools required and estimated hardware utilization.
- What will be the constraints which affect the testing process?

Test Case: A test case is a set of instructions designed to discover a particular type of error or defect in the software system by inducing a failure. The goal of selected test cases is to ensure that there is no error in the program and if there is it then should be immediately depicted. Ideal test casement should contain all inputs to the program. This is often called exhaustive testing.

There are two criteria for the selection of test cases:

- Specifying a criterion for evaluating a set of test cases.
- Generating a set of test cases that satisfy a given criterion.

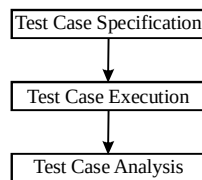


Fig. 7.2 Process of Test Case

1. **Test Case Specification:** Test plan focuses on approach; does not deal with details of testing a unit. Test case specification has to be done separately for each unit. Expected outcome also needs to be specified for each test case. Together the set of test cases should detect most of the defects. Would like the set of test cases to detect any defect, if it exists. The effectiveness and cost of testing depends on the set of test cases. Preparing test case specifications is challenging and time consuming. So for each testing, test case specifications are developed, reviewed, and executed.

2. **Test Case Execution:** Executing test cases may require drivers or stubs to be written; some tests can be automatic, others manual. A separate test procedure document may be prepared.

3. **Test Case Analysis:** Test summary report is often an output – gives a summary of test cases executed, effort, defects found, etc. the task of test case analysis are monitoring of testing effort is important to ensure that sufficient time is spent, computer time also is an indicator of how testing is proceeding, quality control focuses on removing defects and goal of defect prevention is to reduce the defect injection rate in future.

10.6 SOFTWARE VERIFICATION AND VALIDATION

The validation and verification are two different functions in software engineering. We need to understand the difference between them.

Verification: All actions are taken at the end of software development phase to check whether we are building the product right. Verification is done at the end of each phase of software development cycle.

Validation: All actions are taken at the end of software development cycle to check whether we are building the right product. Validation is done at the end of complete development of the product.

These terms are often used interchangeably but following table shows how these terms are different:

Sr.No.	Verification	Validation
1	Verification is aimed to answer question like, Are we building the product right?	Validation is aimed to answer question like, Are we building the right product?
2	Verification ensures that the software system meets all the functionality.	Validation ensures that functionalities meet the intended behavior.
3	Verification takes place first and includes the checking for code, documentation etc.	Validation occurs after verification and mainly involves the checking of the overall product.
4	Verification is done by developers who developed that software.	Validation is done by expert team of testers.
5	Verification is a static activity as it includes the reviews, walkthroughs and inspections to verify that software is correct or not.	Validation has dynamic activities as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify the software.	It is a subjective process and involves subjective decisions on how well the software works.

Table 7.4 Difference between verification and validation

7.7 BLACK BOX TESTING

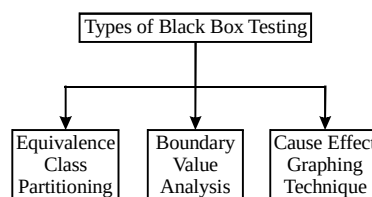
1. Black box testing is also called behavioral testing, focuses on the functional requirements of the software. Black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

2. In black-box testing, the tester only knows the inputs that can be given to the system and what output the system should give. In other words, the basis for deciding test cases in functional testing is the requirements or specifications of the system or module. This form of testing is also called functional or behavioral testing.

3. Black-box testing is not an alternative to white-box techniques. It is a complementary approach that is likely to uncover a different class of errors than white-box methods. It is applied at the later stages of testing. Black-box testing identifies the following kinds of errors:

- Incorrect or missing functions.
- Interface missing or erroneous.
- Errors in data model.
- Errors in access to external data source.

Types of Black-box Testing:



7.3 Types of Black-box Testing

Equivalence class partitioning and boundary value analysis both are test case design strategies in black box testing.

1. **Equivalence Class Partitioning:**

- The domain of the input values partitioned into set of equivalence class. According to equivalence class partitioning, testing the code with one value from an equivalence class is as good as testing the code with any other value from the same equivalence class.
- In this method the input domain data is divided into different equivalence data classes. It is used to reduce the total number of test cases to a finite set of test cases which covers maximum requirements.
- It is a process of taking all possible test cases and placing them into different classes. One test value is picked from each class while testing.
- Example: If we test for an input box which accepts numbers from 1 to 1000 then there is no use in writing thousand test cases for all 1000 valid input numbers and test cases for invalid data. Using equivalence partitioning method, we can divide our test cases into three equivalence classes of some valid and invalid inputs.

2. **Boundary Value Analysis:**

- Boundary value analysis is a next part of equivalence class partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.
- This technique is used to identify errors at boundaries rather than finding those exist in center of input domain. More application errors occur at the boundaries of input domain.
- It is often called stress and negative testing.
- Example: If we test for an input box which accepts numbers from 1 to 1000 then test cases are: one test case for exact boundary values of input domains each means 1 and 1000, one test case for just below boundary value of input domains each means 0 and 999, one test case for just above boundary values of input domains each means 2 and 1001.

3. **Cause Effect Graphing Techniques:**

- The Cause-Effect graph technique restates the requirements specification in terms of logical relationship between the input and output conditions. The causes are the input conditions and effects are the results of those input conditions.
- This technique is used to recognize the root causes, the cause for an exact effect, problem, or outcome.
- In this technique, first we recognize the input conditions (causes) and actions (effect). Then we build up a cause-effect graph and convert cause-effect graph into a decision table. Finally we convert decision table rules to test cases.

Advantages of Black-box Testing

Sr. No	Advantages
1	The test is unbiased because the designer and the tester are independent of each other.
2	The tester does not need knowledge of any specific programming languages.
3	The test is done from the point-of-view of the user, not the designer.
4	Test cases can be designed as soon as the specifications are complete.
5	Well suited and efficient for large code segments.
6	Code Access not required.
7	Clearly separates user's perspective from the developer's perspective through visibly defined roles.
8	Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.

Table 7.5 Advantages of Black-box Testing

Disadvantages of Black-box Testing

Sr. No.	Disadvantage
1	Limited Coverage since only a selected number of test scenarios are actually performed.
2	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
3	Blind Coverage, since the tester cannot target specific code segments or error prone areas.
4	The test cases are difficult to design.

Table 7.6 Disadvantages of Black-box Testing

7.8 WHITE BOX TESTING

White-box testing is also known by other names, such as glass-box testing, structural testing, clear-box testing, open-box testing, logic-driven testing, and path-oriented testing. A complementary approach to functional or black-box testing is called structural or white-box testing. In this approach, test groups must have complete knowledge of the internal structure of the software. We can say structural testing is an approach to testing where the tests are derived from knowledge of the software's structure and implementation. Structural testing is usually applied to relatively small program units, such as subroutines, or the operations associated with an object. As the name implies, the tester can analyze the code and use knowledge about the structure of a component to derive test data. The analysis of the code can be used to find out how many test cases are needed to guarantee that all of the statements in the program are executed at least once during the testing process. It would not be advisable to release software that contains untested statements as the consequence might be disastrous.

The nature of software defects are:

Sr. No.	Nature of software defects in white box testing
1	Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed.
2	We often believe that a logical path is not to be executed when, in fact, it may be executed on a regular basis.
3	Typographical errors are random. When a program is translated into programming language source code, it is likely that some typing errors will occur.

Table 7.7 Nature of Software Defects in White Box Testing

Different forms of White Box Testing

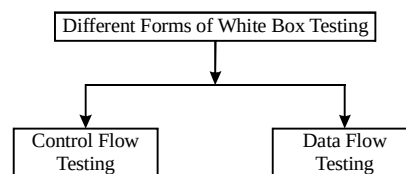


Fig. 7.4 Different forms of White Box Testing

- **Control Flow Testing:** Control flow testing is a structural testing strategy which uses the program's control flow. It is a directed graph where the nodes represent the processing statements like definition, computation and predicates while the edges represent the flow of control between processing statements. It is more effective for unstructured code rather than structured code. But it cannot catch initialization and specification mistakes. To examine the patterns, we need to construct a control flow graph of the code.

- **Data Flow Based Testing:** Data flow testing is a technique used to detect improper use of data in a program. It is a white box testing technique that can be used to detect improper use of data values due to coding errors. Data-flow testing uses the control flow graph to explore the unreasonable things that can happen to data. Data-flow testing is the set of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of data objects. Data-

flow testing strategies span the gap between all paths and branch testing. Data-flow testing monitors the lifecycle of a piece of data and identifies potential bugs by examining the patterns in which that piece of data is used.

Reasons Why White-box Testing is performed

Sr. No.	Reasons
1	All paths in a process are correctly operational.
2	All logical decisions are executed with true and false conditions.
3	All loops are executed with their limit values tested.

Table 7. 8 Reasons Why White-box Testing is Performed

Types of White Box Testing

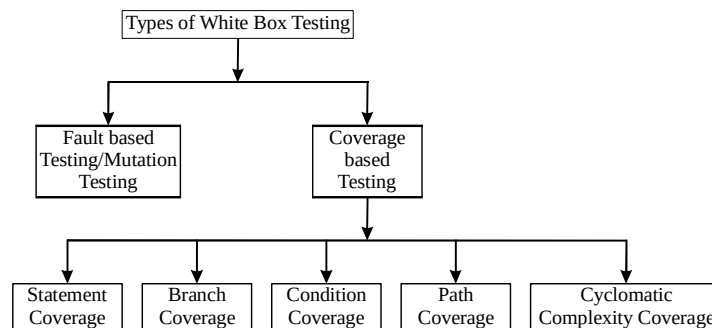


Fig. 7.5 Types of White Box Testing

1. **Fault Based Testing:** It is used to detect certain types of faults. Its goal is to determine the absence of faults rather than the goal of “finding errors”. The focus is on faults rather than errors. It is not a manual testing. The example of fault based testing is mutation testing.

2. **Coverage Based Testing:** A coverage based testing strategy aims to execute certain program elements to discover failures. It is of four types:

- **Statement Coverage:** It aims to design test cases so as to execute all the statements of a program. In this strategy, every statement in a program is to be executed at least once. The disadvantage of this strategy is that after executing a statement for one input value once, it assumes that it behave properly but there is no guarantee that it will behave correctly for all input values.

- **Branch Coverage:** It is also known as edge testing. It is used to design test cases in such a way that it assume true and false value in return. Branch coverage testing is stronger than statement coverage testing because branch coverage ensures statement coverage but statement coverage does not ensure branch coverage.

- **Condition Coverage:** In this testing strategy, test cases are designed to make each component of a conditional expression to assume both true and false values. Condition coverage testing is a stronger testing strategy than branch coverage and Branch testing is a simplified condition testing.

- **Path Coverage:** It design test cases in such a manner that all linearly independent paths in the program are executed at least once. To understand each linearly independent path, we need to understand the control flow graph (CFG). A control flow graph is a directed graph which consists of set of nodes and edges (N, E). It describes how the control flows through the program.

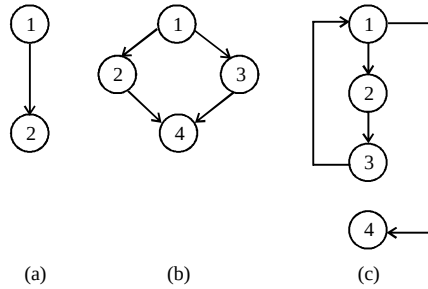


Fig. 7.6 Examples of Path Coverage

- **Cyclomatic Complex Graph:** It tests the complexity of the program. It also estimates the amount of effort required to understand the code. The Cyclomatic complexity is defines as:

$$CC = E - N + P$$

CC = Cyclomatic Complexity

E = Number of Edges

N = Number of Nodes

P = Number of Connected Components

Example:

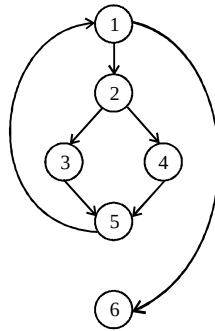


Fig. 7.7 Examples of Cyclomatic Complexity

If number of edges = 7 and number of nodes = 6, then cyclomatic complexity is:

$$CC = 7 - 6 + 2 \Rightarrow CC = 3$$

Cyclomatic complexity helps in estimation of complexity of code, testing effort and program reliability.

Advantages of White-Box Testing:

Sr. No.	Advantages
1	As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.
2	It helps in optimizing the code.
3	Extra lines of code can be removed which can bring in hidden defects.
4	Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.

Table 7.9 Advantages of White Box Testing

Disadvantages of White-Box Testing:

Sr. No.	Disadvantages
1	Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.
2	Sometimes it is impossible to look into every nook and corner to find out hidden error s that may create problems as many paths will go untested.
3	It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.

Table 7.10 Disadvantages of White Box Testing

7.9 GREY BOX TESTING

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term “the more you know the better” carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application’s user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.

Advantages:

1. Offers combined benefits of black box and white box testing wherever possible.
2. Grey box testers don’t rely on the source code; instead they rely on interface definition and functional specifications.
3. Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication protocols and data type handling.
4. The test is done from the point of view of the user and not the designer.

Disadvantages:

Since the access to source code is not available, the ability to go over the code and test coverage is limited. The tests can be redundant if the software designer has already run a test case. Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.

7.10 DIFFERENCE BETWEEN BLACK BOX TESTING, GREY BOX TESTING AND WHITE BOX TESTING

Sr. No.	Black Box Testing	Grey Box Testing	White Box Testing
1.	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2.	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3.	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4.	Testing is based on external expectations Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5.	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6.	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing

Table 7.11 Difference Between Black Box Testing, Grey Box Testing and White Box Testing

7.11 LEVELS OF TESTING

There are four levels of testing, i.e., four individual modules in the entire software system.

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance testing

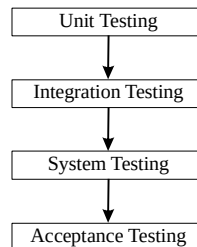


Fig 7. 8 Level of Testing

7.11.1 Unit Testing

The unit test is the lowest level of testing performed during software development, where individual units of software are tested in isolation from other parts of a program. The individual components are tested to ensure that they work correctly. It focuses on the smallest unit of software design where each component is tested independently without other system components. There are number of reasons to do unit testing rather than testing the entire product:

Reasons to perform Unit Testing:

- The size of a single unit is small that is easy to locate error in it.
- The unit is small enough that we can attempt as much as test cases.
- The interactions of multiple errors in it are eliminated.

Benefits of Unit testing:

The unit testing has the following benefits:

1. **Increases confidence:** The unit testing increases confidence in changing/maintaining code. If good unit tests are written and if they are run every time any code is changed, the likelihood of any defects due to the change being promptly caught is very high. If unit testing is not in place, the most one can do is hope for the best and wait till the test results at higher levels of testing are out. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.
2. **Reusability:** The codes are more reusable. In order to make unit testing possible, codes need to be modular.
3. **Fastest development:** The development is faster as the effort required to find and fix defects found during unit testing is easy as comparison to finding defects during system testing or acceptance testing.
4. **Fewer efforts:** The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.
5. **Easy Debugging:** Due to unit testing the debugging is easy. When a test fails, only the latest changes need to be debugged.
6. **Error discovery:** Unit testing can discover error like Comparison of different data types, Incorrect logical operators or precedence, Expectation of equality when precision error makes equality unlikely, Incorrect comparison of variables, Improper loop termination, Failure to exit when divergent iteration is encountered, Improperly modified loop variables

Unit-test Procedure:

In unit testing, a module is tested under well-defined environment which includes all relevant code that is required to execute the module. The following diagram shows environment:

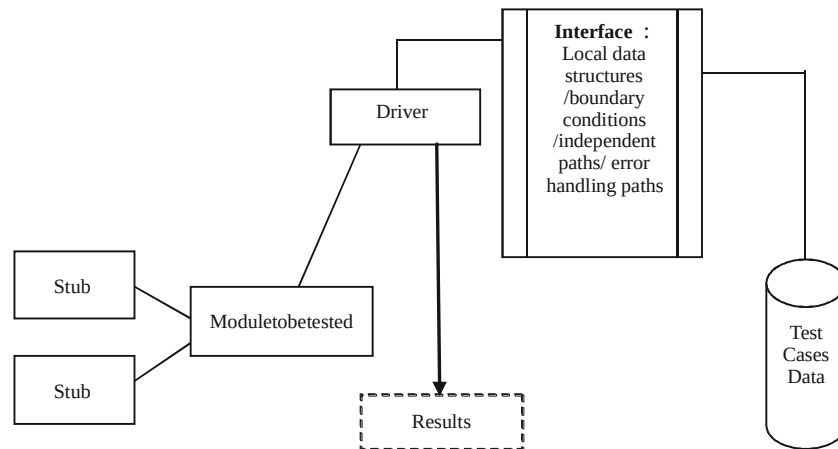


Fig. 7.9 Unit-test Environment

In the unit test environment, there are four components namely stub, driver, interface, test cases.

Stubs: The Stubs serve to replace modules that are subordinate to the component to be tested.

Driver: A driver is nothing more than a “main program” that accepts test-case data, passes such data to the component (to be tested), and prints relevant results.

Interface: The interface manages tasks like integration of local data structure, keep check on boundary condition and possible path to execute the code. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that the module operates properly at the boundaries established to limit. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.

Test case data: The test case data is collection of all possible input that can be observed during the testing phase of a particular unit.

7.11.2 Integration Testing

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. In this testing many unit-tested modules are combined into subsystems, which are then tested. The goal here is to see if the modules can be integrated properly.

Objective of Integration Testing:

The primary objective of integration testing is to test the module interfaces in order to ensure that there are no errors in the parameter passing, when one module invokes another module. During integration testing, different modules of a system are integrated in a planned manner using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested.

Approaches to Integration Testing:

The various approaches used for integration testing are:

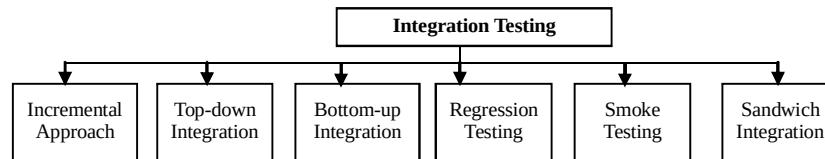


Fig. 7.10 Approaches to integration testing

- **Incremental Approach:** According to the incremental approach, testing starts with first combine only two components together and performs testing on them. If there are some error then removes them first, otherwise combine another component to it and then test again, and so on until the whole system is developed. The following logical diagram shows the same process:

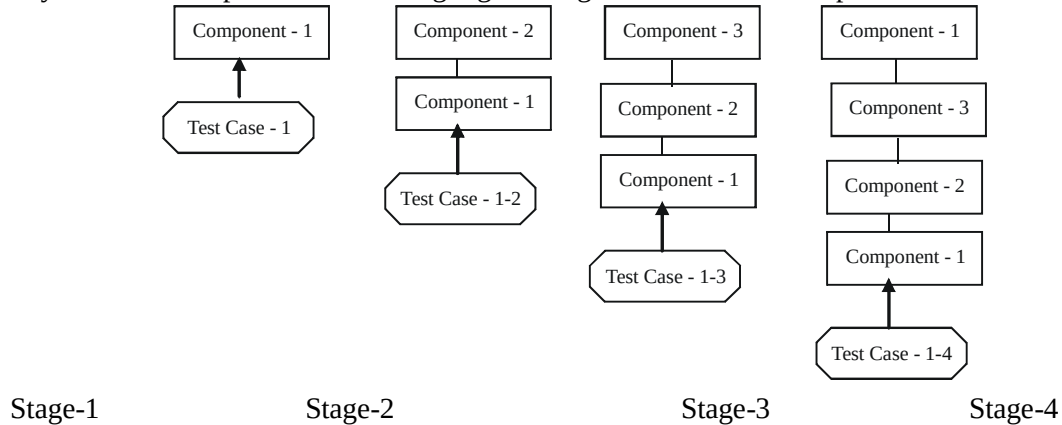


Fig. 7.11 Incremental approach for testing

According to Figure, in incremented testing is performed in four stages where in stage1, component1 is tested if it is working correctly then it is integrated with componets2 and gain test cases1 to 2 are used for testing. If these components are corrected or error free then module componets3 is integrated, i.e. stage3 and then tests case 1 to 3 applied on all the components. The process is repeated till whole system is not integrated and tested.

- **Top-Down Integration Testing**

Top-down integration testing is an incremental approach to construction of program structures. Modules are integrated by moving downward through the control hierarchy beginning with the main control module.

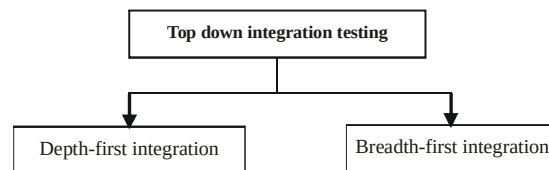
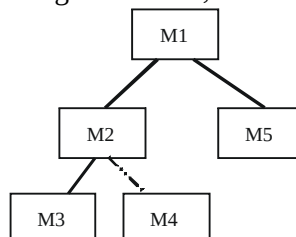


Fig. 7.12 Types of Top down Integration Testing

Depth-first integration: According to depth first integration, we integrate all components on a major control path of the structure. Let us consider example: For example, M1, M2, and M3 would be integrated first. Next, M4, M6 would be integrated. Then, the left side depth control paths are built.



Breadth-first integration: According to breadth first integration all components directly subordinate at each level, moving across the structure horizontally will be incorporated as shown in figure. The components M2, M5 will be integrated first. The next control level incorporate M3, M4. Figure: Top-down integration testing

- **Bottom up Integration Testing.** The bottom-up integration testing begins construction and testing with the components at the lowest level in the program structure. Then low level tested components are integrated and again tested as group. A bottom-up integration strategy may be implemented with the following steps:

1. Low-level components are combined into clusters that perform specific software sub functions.
2. A driver is written to coordinate test case input and output.
3. The cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

The bottom up integration is demonstrated in the following diagram:

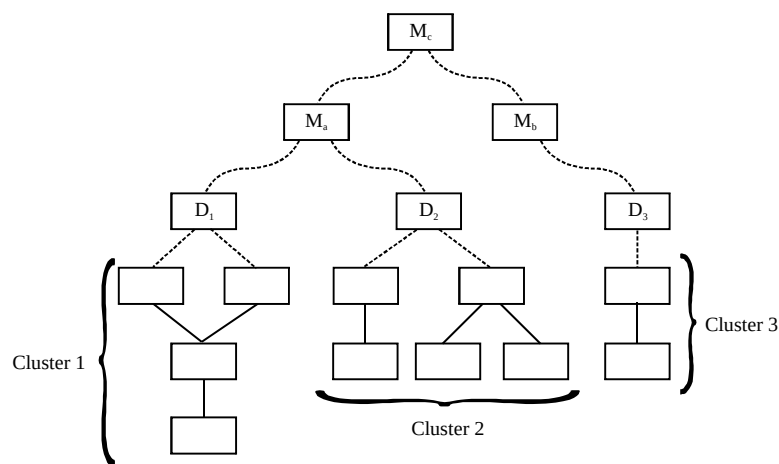


Fig. 7. 13 Bottom-up Integration Testing

The components are combined to form clusters 1, 2, 3. Every cluster is tested using a driver and components in clusters 1 and 2 are subordinate to Ma. Drivers D1 and D2 are removed and the clusters are interfaced directly to Ma. Similarly, driver D3 for cluster 3 is removed prior to integration with module Mb. Both Ma and Mb will ultimately be integrated with component Mc and so forth.

- **Regression Testing:** Regression testing is the activity that helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors. The regression test suite contains three different classes of test cases:

- Additional tests that focus on software functions.
- A representative sample of tests that will exercise all software functions.
- Tests that focus on the software components that have been changed.

- **Smoke Testing:** Smoke testing is an integration testing approach that is commonly used when “shrink-wrapped” software products are developed. Smoke testing is characterized as a rolling integration approach because the software is rebuilt with new components and testing. Smoke testing encompasses the following activities:

- Software components that have been translated into code are integrated into a “build.” A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.

- A series of tests is designed to expose errors that will keep the build from properly performing its functions. The build is integrated with other builds and the entire product (in its current form) is smoke tested daily.
- **Sandwich Integration Testing.** Sandwich integration testing is the combination of both the top-down and bottom-up approach. So, it is also called mixed integration testing. In it, the whole system is divided into three layers, just like a sandwich: the target is in the middle and one layer is above the target and one is below the target.

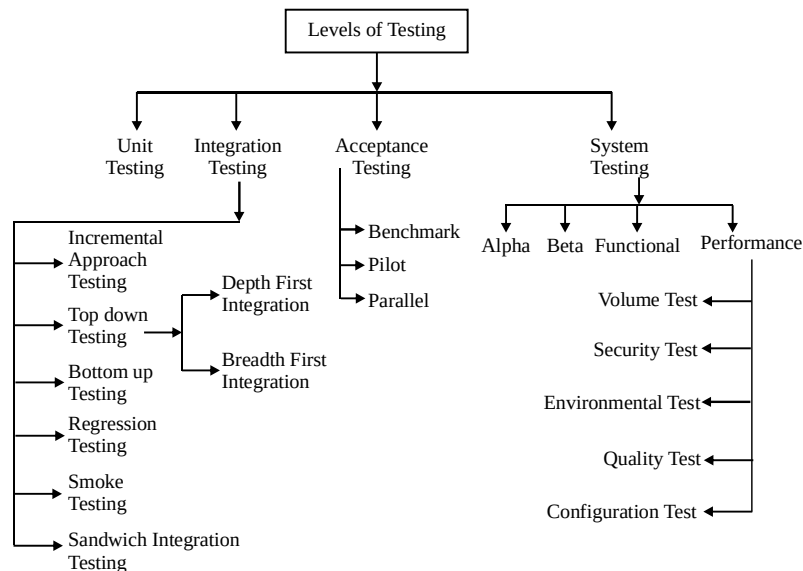


Fig 7.14 Levels of Testing in Detail

7.11.3 System Testing

Once all the components are integrated, the application as a whole is tested thoroughly to see that it meets Quality Standards. This type of testing is performed by a specialized testing team. It is also concerned with validating that the system meets its functional and non-functional requirements. There are essentially three main kinds of system testing:

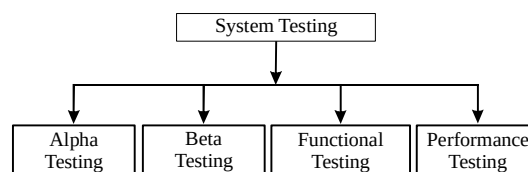


Fig 7.15 Classification of System Testing

- **Alpha Testing:** Alpha testing refers to the system testing carried out by the test team within the development organization.

The alpha test is conducted at the developer's site by the customer under the project team's guidance. In this test, users test the software on the development platform and point out errors for correction. However, the alpha test, because a few users on the development platform conduct it, has limited ability to expose errors and correct them. Alpha tests are conducted in a controlled environment. It is a simulation of real-life usage. Once the alpha test is complete, the software product is ready for transition to the customer site for implementation and development.

- **Beta Testing:** Beta testing is the system testing performed by a selected group of friendly customers.

If the system is complex, the software is not taken for implementation directly. It is installed and all users are asked to use the software in testing mode; this is not live usage. This is called the beta test. Beta tests are conducted at the customer site in an environment where the software is exposed to a number of users.

The developer may or may not be present while the software is in use. So, beta testing is a real-life software experience without actual implementation. In this test, end users record their observations, mistakes, errors, and so on and report them periodically.

In a beta test, the user may suggest a modification, a major change, or a deviation. The development has to examine the proposed change and put it into the change management system for a smooth change from just developed software to a revised, better software. It is standard practice to put all such changes in subsequent version releases.

- **Functional Testing:** *Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.*

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. There are five steps that are involved when testing an application for functionality.

1. The determination of the functionality that the intended application is meant to perform.
2. The creation of test data based on the specifications of the application.
3. The output based on the test data and the specifications of the application.
4. The writing of Test Scenarios and the execution of test cases.
5. The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

- **Performance Testing:** Performance testing checks the performance objectives set by the customer. It is also known as nonfunctional testing. It measures the accessibility of data, accuracy of the output, speed of response etc. The following are the types of performance testing:

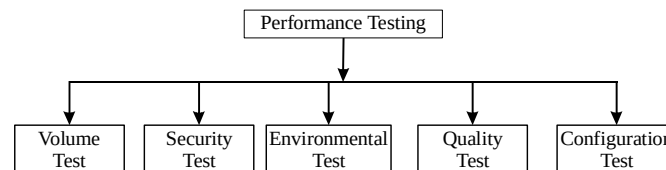


Fig. 7.16 Types of Performance Testing

Volume Test: It address the handling of large amount of data in the system.

Security Test: It ensures the security requirements of the system.

Environmental Test: This test ensures performance of system under the tolerance for heat, motion, portability, moisture and other environmental situations.

Quality Test:It ensures the reliability, availability and maintainability of the system.

Configuration Test: It ensures the various software and hardware configurations explained in the requirements.

7.11.4 Acceptance Testing

Acceptance testing is the system testing performed by the customer to determine whether to accept or reject the delivery of the system.

When customer software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements. Conducted by the end-user rather than the software engineers, an acceptance test can range from an informal 'test drive' to a planned and systematically

executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. Acceptance tests are written, conducted and evaluated by the customers.

Types of acceptance tests:

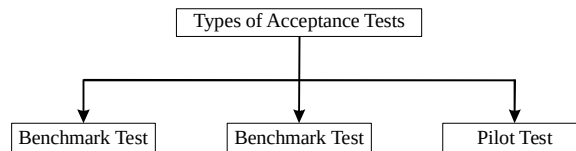


Fig. 7.17 Types of Acceptance Tests

- **Benchmark Test:** These tests are performed when a user has special requirements. These tests are performed by actual user or a special team.
- **Pilot Test:** Pilot tests rely on all the functions of everyday working of the system. User exercise the system as if it had been installed permanently.
- **Parallel Test:** In this test, the new system installs and operates in parallel with the previous version. It builds the confidence of user in new system with the old system.

BACHELOR OF COMPUTER APPLICATIONS (BCA)
BCA-4-01T: SOFTWARE ENGINEERING

UNIT 8: SOFTWARE MAINTENANCE AND RELIABILITY

8.1 INTRODUCTION

8.2 SOFTWARE MAINTENANCE

8.2.1 OBJECTIVES OF SOFTWARE MAINTENANCE

8.2.2 CATEGORIES OF SOFTWARE MAINTENANCE

8.2.3 COMPONENTS OF SOFTWARE MAINTENANCE PROCESS

8.2.4 SOFTWARE MAINTENANCE PROCESS

8.2.5 FACTORS AFFECTING THE SOFTWARE MAINTENANCE

8.3 SOFTWARE RELIABILITY

8.3.1 FACTORS AFFECTING SOFTWARE RELIABILITY

8.3.2 COMPARISON OF SOFTWARE RELIABILITY AND HARDWARE RELIABILITY

8.3.3 USES OF RELIABILITY STUDIES

8.3.4 SOFTWARE RELIABILITY METRICS

8.3.5 SOFTWARE RELIABILITY GROWTH MODELS

8.1 INTRODUCTION

Software maintenance is the activity associated with keeping an operational computer system continuously in tune with the requirements of users and data processing operations. It process is expensive and risky and is very challenging. Maintenance may be classified into the four categories as follows: Corrective, Adaptive, Perfective, and Preventive. The term software reliability is defined as the probability of failure-free operation of a computer program in a specified environment for a specified time. The detailed discussion about categories of software maintenance along with different components of software maintenance procedures are discussed with appropriate examples.

8.2 SOFTWARE MAINTENANCE

1. After the complete development of software product, software is handed over to the client. It client requires any changes in the software product after the delivery then it is turned as software maintenance. Software maintenance is the most difficult aspect of software development.

Definition

According to **IEEE**: *“Software maintenance is a process of modification in a software product after delivery to correct faults, or to improve performance so that software product can adopt the modified environment.”*

2. Maximum time is spent on software maintenance as compared to the development of any phase of the software product. So it is very important to kept maintenance in mind throughout the software development.

3. Once an error has been located, the maintenance programmer resolves the errors without adding any error (regression errors). Detailed documentation helps the maintenance programmer in correcting the errors.
4. After receiving a maintenance request, the first step is to identify what type of error is there and what type of maintenance is required.
5. After correcting an error, the fix is tested again to ensure the correct functionality of the software. It also ensure that there should be no other side effects.
6. There are number of software life cycle models which ignore the software maintenance phase and do not provide any adequate information for integration of maintenance into development process.
7. Software development and software maintenance are two processes of software engineering. Software maintenance is based on the information gained from software development. Without this information, software maintenance is very difficult process designing, coding, testing etc.

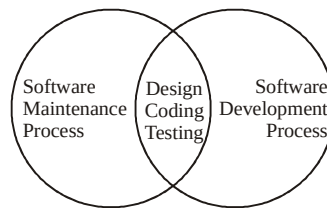


Fig. 8.1 Relationship between Software Maintenance Process & Software Development Process

8. Software maintenance costs are underestimated during the design and implementation phase. But in reality software maintenance costs are the greatest costs.
9. One of the issue regarding maintenance is that software engineers have very poor image of maintenance. They it is unimportant and easy task. That is why organizations allocated this maintenance task into very less skilled and inexperienced staff which results in very high maintenance cost and time.
10. The maintenance cost and time can be reduced by involving experienced and skilled staff early in the software process.
11. In short, software maintenance means modification in a software product after delivery. It mostly occurs at customer's site. Software maintenance include error detection, error correction and enhancement of capabilities.

8.2.1 Objectives of Software Maintenance

Software maintenance is a set of activities performed in a software product after delivery. Example of software maintenance are changes in code documentation manuals or any part of the product etc. Software maintenance has many objectives which are as follows:

1. Software maintenance enhances the performance of the software.
2. It correct various errors occur during the functionality of the software.
3. It makes changes in the software so that it can adopt the changes in the environment.
4. Software maintenance also update the software product to avoid future problems.

Objectives of Software Maintenance

- Enhances the performance
- Correct errors
- Adopt the modified environment
- Avoid future problems
- Satisfies the customer needs
- Delete obsolete features

Fig. 8.2 Objectives of Software Maintenance

5. Maintenance process satisfies the customer needs. After maintenance, software starts working according to user requirements.
6. Software maintenance is used to correct program errors, to add new capabilities into the software to delete obsolete features and to improve performance of the software.

8.2.2 Categories of Software Maintenance

Software maintenance is categorized into four types which are as follows:

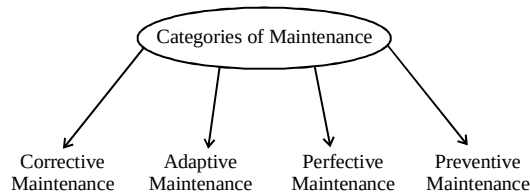


Fig. 8.3 Categories of Maintenance

I. Corrective Maintenance

1. Corrective maintenance is necessary to resolve the bugs occur during the usage of a software product. Bugs can occur at any stage during the development of software for example—bugs can occur during the specifications, design, coding and testing etc.
2. Corrective maintenance changes the software to correct the errors. Some errors are caused by data processing error and system performance errors.
3. An error can result from design errors, logic errors and coding errors which are as follows:
 - **Designing errors:** These errors occurs when change are made in the software. These changes are incorrect, incomplete or misunderstood and arises errors.
 - **Logic errors:** These errors occurs when invalid tests, incomplete test data of design specification and incorrect implementations are made in the software.
 - **Coding errors:** These errors occurs when incorrect use of source code logic and wrong implementation of detailed logic design are made in the software.

II. Adaptive Maintenance

1. Adaptive maintenance makes changes in the software so that it can work in new environment (new platforms) without any error. For example—porting to a new compiler, operating system and/or hardware.
2. The term new environment refers to the totality of all conditions/influences that act from outside on the software.
3. Adaptive maintenance in the software is not the need of the client. It is the need of external environment in which software works.

III. Perfective Maintenance

1. Perfective maintenance needs time and money. It improves the efficiency, effectiveness and maintainability of the software.
2. In perfective maintenance, additions or enhancements are made in the software to improve its functionality and quality.
3. The perfective maintenance is requested by software engineer and the client. The software engineer needs perfective maintenance to improve the status of the product in market by improving its quality. The client needs perfective maintenance to meet the new requirements.

IV. Preventive Maintenance

1. Preventive maintenance performs the activities/operations on the software to update it in anticipation of any future problems.
2. It performs modifications in a software after delivery, to improve the effectiveness, efficiency and performance so that software does not face any problem (errors) in the future.
3. Preventive Maintenance is basically a modification in the software after the delivery to detect and correct the faults. It ensures the safety of the software.

8.2.3 Components of Software Maintenance Process

The components mean those parts which play an important role in software maintenance. The important components of the software maintenance process are people, tasks and knowledge of the software. These components are inter-related and interdependent on each other.

1. **People:** It is the most important component of the software maintenance process. This component covers all those persons which are the part of the maintenance process directly or indirectly. It includes users, development staff and maintenance staff.

- Users are those people for whom software is developed and maintained.
- Development staff is a team of those people who take responsibility for the successful development of the software product.
- Maintenance staff includes those people who participated actively in the software product after the complete development of the software to make it up to date.

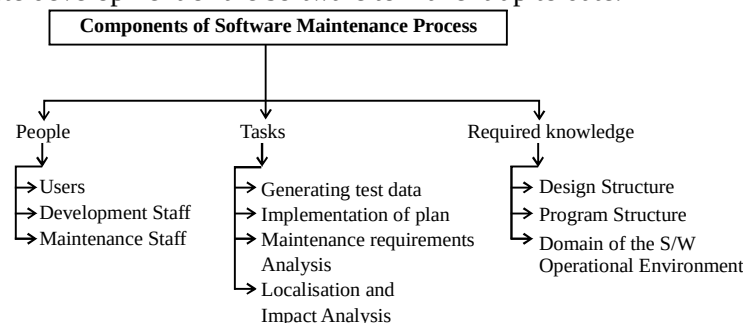


Fig. 8.4 Components of Software Maintenance Process

2. **Tasks:** Tasks mean those activities or operations which are performed during a successful maintenance process. During the maintenance process, various tasks like implementation plan, maintenance requirements analysis, localization and impact analysis etc. are performed.

- Implementation plan provides the step by step maintenance requirement analysis and the need of maintenance requirement is checked.
- It is used to examine the requirements of the maintenance. Impact analysis includes the detailed study of software product after the maintenance process. Performance, efficiency and effectiveness of software product is examined after the successful implementation of maintenance process.

3. **Required knowledge:** This component includes the structured and unstructured knowledge for the software maintenance. Maintenance staff should have the knowledge of the design and program structure of the software product. Maintenance staff should have complete knowledge of domain of the software along with the operational environments.

8.2.4 Software Maintenance Process

The maintenance process has five steps which are as follows:

1. **Understand the existing programs:** In this step, before modification, existing programs are studied for complete understanding. Important attributes which are helpful in better understanding of

existing programs are: less complexity and proper documentation. In this step, objectives are also studied in the detailed manner.

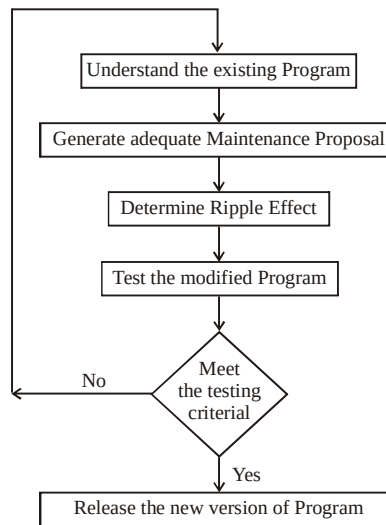


Fig. 8.5 Steps of Software Maintenance Process

2. **Generate adequate maintenance proposal:** In this step, maintenance team generates and propose the adequate maintenance proposal to satisfy the maintenance objectives.

3. **Determine ripple effect:**

- Ripple effect determines the effect of the change (modification) on the system.
- Ripple effect is logical/functional in nature. It concerns the performance of the program.
- The ripple effect is affected by the consequence of a program modification i.e. the stability of the program.
- Program stability is basically a resistance to the amplification of changes in the program.
- Regression testing is performed to ensure that unchanged code is still working efficiently after making the new changes in the software product.

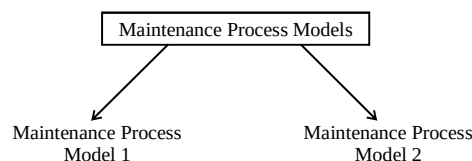
4. **Test the modified program:** In this step, modified program is tested to ensure the reliability level. Modified program should have at least same reliability level as before. Cost effective testing techniques are also performed (during the maintenance process) to measure the required effort to test the modified program, under well-defined testing conditions.

5. **Release the new version of program:** After meeting the testing conditions, modified programs with new versions are ready to install/release.

Maintenance Process Models

No single maintenance process model is suitable for all types of maintenance projects. Maintenance process activities are different for different maintenance projects.

There are two bread categories of maintenance process models are proposed.



Model 1 preferred for projects where small re-work is required. In this model, the code is changed directly and the changes are reflected in the relevant documents later.	Model 2 preferred for projects where amount of rework required is significant. In this model, reverse engineering cycle is followed by a forward engineering cycle. This approach is also known as software reengineering.

Fig. 8.6 Maintenance Process Models

1. **Maintenance Process Model 1:** The maintenance process model 1 starts with gathering the requirements for change. This is the first step of Model 1. In this step, all the change requirements are gathered in a detailed manners. In this step 2, these change requirements are analyzed and studied from different angles. In this step 3, after analyzing the change requirements, efficient strategies are formulated and adopted for code change. At this stage, few members of original development team goes a long way in reducing the cycle time, for the projects involving unstructured and inadequate documented code.

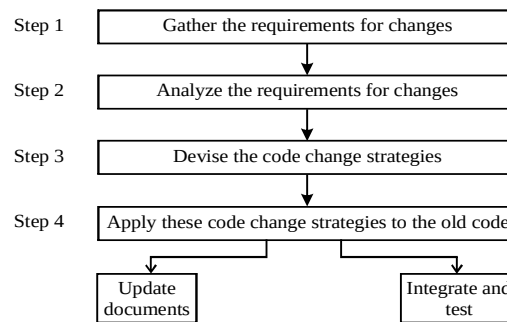


Fig. 8.7 Maintenance Process Model 1

In this step 4, apply the formulated and adopted code strategies to the old code. Maintenance engineers study the working of the old system and compare the working of their modified system with the old system.

In the end, debugging of the new modified system become easy as after the comparison of the old and new modified system, bugs can be easily localize. In the model 1, the output is update documents and integrate and test the modified program.

2. **Maintenance Process Model 2**

- Maintenance Process Model 2 is basically suitable for those projects where amount of rework is required. In this model reverse engineering cycle is followed by a forward engineering cycle which is known as software re-engineering.
- During the reverse engineering, the old coding is analyzed to extract the module specification. The module specification is further analyzed for producing the designing part. The design is further analyzed to produce the original requirement specification. This process is known as reverse re-engineering because analyzing process is done in reverse order means from bottom to top stages.
- After analyzing the original requirement specifications, change requirements are applied to the original requirement specifications, to produce the new requirement specification. After arriving at the new requirement specification, forward engineering approach is used to produce a new code.

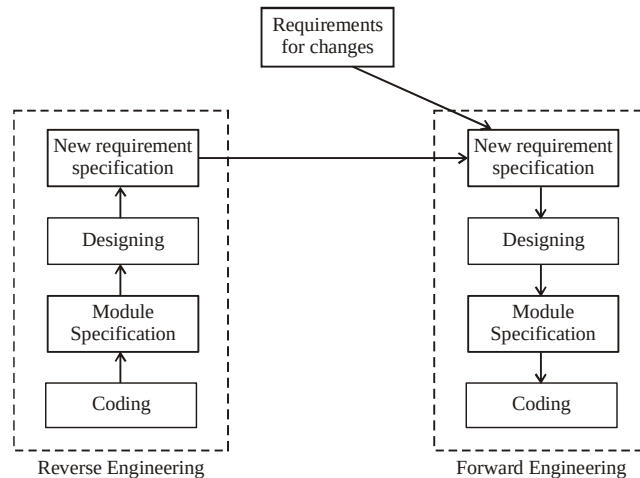


Fig. 8.8 Maintenance Process Model 2

- During the forward engineering approach, new requirement specifications are considered. At the designing, module specification and coding stages, a substantial reuse is made from the reverse engineered products. This approach produces good documentation, more structured design as compared to the original product.

Comparison between Model 1 and Model 2

1. The maintenance process model 2 gives efficient outputs, efficient designs and adequate documentation as compared to maintenance process model 1.
2. The maintenance process model 2 is cost lies than the process model 1.
3. The maintenance process model 2 is preferred only when amount of rework is very high. If amount of rework is not more than 20%, then we use model 1.

8.2.5 Factors affecting the Software Maintenance

There are broadly two categories of factors which affects the software maintenance.

I. **Technical Factors:** The technical factors of software maintenance are as follows:

1. **Programming Language:** Programming language plays important role in the software maintenance. Programs written in the high level language are easier to understand and maintained than the programs written in low level languages.
2. **Program Testing:** If a program is adequately validated and tested during the design phase, then there are less chances of occurring errors in the software product. If development staff spends maximum effort and time on the design testing then maintenance cost is reduced because of fewer errors in the program.

Maximum time and effort in design phase = Less maintenance cost

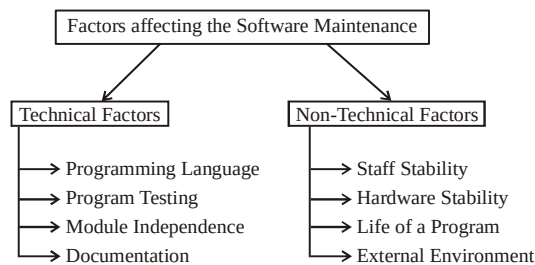


Fig. 8.9 Factors affecting the Software Maintenance

3. **Module Independence:** Module independence means modification of one program unit of a system without affecting any program unit of a system. If one module is being modified, then there should be no effect on other modules of software product.
4. **Documentation:** Documentation is another important factor of software maintenance. If a program is clearly, completely and adequately documented, then it provides the complete understanding of the program.
- II. **Non-Technical Factors:** Then non-technical factors of software maintenance are as follows:
 1. **Staff Stability:** The stability of staff effects the maintenance cost.
 More stable staff = less maintenance cost
 If an original writer of a program understands and changes a program, then it will be very easy for him to maintain that program and becomes very beneficial for the development company in terms of cost and time. It is very difficult for individual to understand and changes in a program written by another person (a new programmer).
 2. **Hardware Stability:** The program must be modified to use new hardware. The new hardware replaces the old (obsolete) equipment. Sometimes a program is designed for a particular hardware configuration and the chances of the change of that particular hardware configuration is very less during the program's whole life. In this situation there is no maintenance cost, but it is rarest case.
 3. **Life of a Program:** As the life of a program increases, the maintenance cost increases. The life of a program depends upon the application used. When the application becomes obsolete, program becomes obsolete and maintenance cost increases very rapidly because of conversion of original obsolete hardware into new hardware.
 Life of Program = Maintenance cost and time
 4. **External Environment:** Sometimes, program depends upon the external environments. According to the external environments, program needs changes and maintenance cost increases. Whole maintenance process and maintenance cost depends upon the external environment.
 Changes in external environment $\propto$ Higher maintenance cost and time.

8.3 SOFTWARE RELIABILITY

1. Software reliability is an important concern for users. Software reliability is an attribute of software quality and it is independent from time.
2. More the software reliability, higher the software quality.
3. If software reliability increases, then the performance usability, functionality and stability of software increases.
4. According to **IEEE**: *"Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment."*
5. Reliability of a software product denotes the trust-worthiness, software reliability refers to as the probability of the product working correctly over a given time duration.
6. Sometimes, it is very difficult to measure accurately the reliability of a software product because of following two reasons:
 - (i) Reliability is observer dependent. It depends upon the observer's new point.
 - (ii) Reliability of a software product keeps changing as errors are detected and fixed. Because of frequent changing nature of software product, problems occur during the accurate measurement of the reliability of the software product.

8.3.1 Factors affecting Software Reliability

There are many factors which affects the growth of software reliability. These factors increases or decreases the growth of software reliability. These factors are as follows:

1. **Number of Errors:** Software reliability depends upon the number. of errors in the software product. Reliability is inversely proportional to the number of errors (defects) occur in the software product.

$$\text{Reliability} \propto \frac{1}{\text{No. of Errors}}$$

Reliability increases when no. of errors decreases in a particular software product. Reliability decreases when no. of errors increases in a particular software product.

2. **Exact Location of Errors:** Software reliability increases when maintenance staff finds out the exact location of error. If exact location of error is finds out, then it becomes very easy for maintenance staff to remove that error and enhance the reliability and quality of a particular software product.
3. **How Product is used:** Software reliability also depends upon the way in which product is being used. If proper and correct way is used then it increases the reliability of a particular software product.
4. **Time:** Software reliability is not a function of time and different from hardware failure. With the passage of time, hardware components become old and obsolete. Hardware failure parts can be replaced with rate of hardware reliability increases. But in the case of software reliability, software cannot wear out. It can be upgraded intentionally by identifying and resolving errors and failure rate decreases. But when the software become obsolete, no more error correction occurs and failure rate remains unchanged.

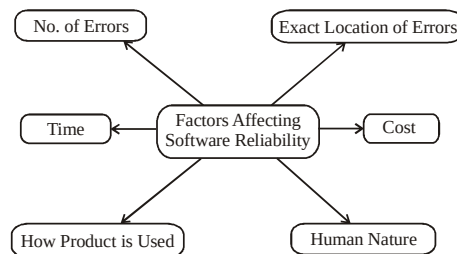


Fig. 8.10 Factors affecting Software Reliability

5. **Cost:** Software reliability depends on cost. Testing is a costly process. When more testing is performed on any software product to improve its reliability, cost automatically increases.
6. **Human Nature:** Software reliability also depends upon the human nature and observer's view point. Level of reliability varies with the different nature of different people.

8.3.2 Comparison of Software Reliability and Hardware Reliability

Measuring the software reliability is more difficult as compared to measuring the hardware reliability. The difference between hardware reliability and software reliability are as follows:

Sr. No	Hardware Reliability	Software Reliability
1	Hardware reliability is concerned with stability.	Software reliability is concerned with reliability growth.
2	Hardware components fail because of wear and tear.	Software components fail because of bugs
3	To fix a hardware fault, one has to repair or replace the failed parts.	To fix a software fault, one has to fix the bug, code is changed to fix the bug.
4	When the hardware failure is repaired its reliability would be maintained at the level that existed before the failure occurred.	When the software failure is repaired, its reliability may increase or decrease.
5	Hardware reliability is a function of time and different from software failure.	Software reliability is not a function of time and different from hardware failure.

Table 8.1 Comparison of Software Reliability and Hardware Reliability

8.3.3 Uses of Reliability Studies

1. It estimate and predict the reliability behavior of software during its development and operation.
2. It answer the various question like what is the expected duration between successive failures.
3. It helps software engineering methodologies to develop the code.
4. It use formal methods during code development.
5. It compares the software engineering technologies in terms of cost and quality.
6. It aims at fault-free performance of software systems.
7. It concerns itself with how well the software functions to meet the requirements of the user.
8. It measures the progress of system testing.
9. It gives us a better insight into the development processes.

8.3.4 Software Reliability Metrics

Software reliability metrics are used to express quantitatively the reliability of the software product. No single metric is present which explains the reliability of all software products. Different software products may have different reliability requirements. The following are some reliability metrics:

1. **Rate of Occurrence of Failure (ROCOF):** It measures the frequency of the occurrences of failure. ROCOF can be obtained by observing the operational behavior of a software product over a specified time interval.

$$\text{ROCOF} = \frac{\text{Total number of observed failures}}{\text{Duration of observation}}$$

2. **Mean Time to Failure (MTTF):** MTTF is mean time between two successive failures. MTTF is measured by recording the failure data for n failures.

$$\text{MTTF} = \sum_{i=1}^n \frac{t_{i+1} - t_i}{(n-1)}$$

3. **Mean Time to Repair (MTTR):** After the occurrence of failure, some time is required to fix the error to solve that failure.

MTTF is the average time to track and fix the errors.

4. **Mean Time between Failure (MTBF):** MTBF metric is a combination of MTTF and MTTR
 $\text{MTBF} = \text{MTTF} + \text{MTTR}$

5. **Availability (Avail):** It is a measure of how likely would the system be available for use over a given period of time. Availability is a measure of time during the system is available over long time periods.

$$\text{Availability} = \frac{\text{MTBF}}{(\text{MTBF} + \text{MTTR})}$$

8.3.5 Software Reliability Growth Models

A software reliability growth model is a mathematical model which helps in detection of errors, repair those errors and improves the software reliability. It also provides information when to stop testing to attain given reliability growth level.

The two software reliability growth models are as follows:

1. Jelinski and Moranda Model
2. Littlewood Model

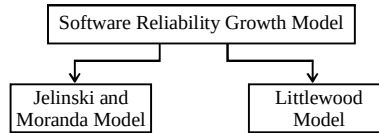


Fig. 8.11 Software Reliability Growth Models

1. **Jelinski and Moranda Model:** It is the simplest reliability growth model. This model assumes that reliability increases each time when an error is detected and repaired. Reliability is directly proportional to error detection and fixing.

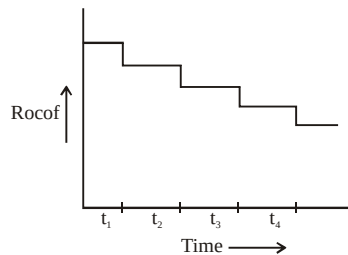


Fig. 8.12 Jelinski and Moranda Model

2. **Littlewood Model:** It is the complex reliability growth model. This model assumes that there is a chance of introducing additional errors when an error is detected and repaired.

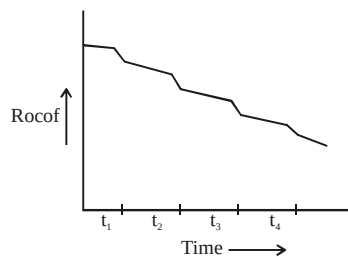


Fig. 8.13 Littlewood Model



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

**The Motto of the University
(SEWA)**

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



**Bachelor of Computer Applications (BCA)
Course Name: Computer Networks
Course Code: BCA-4-02T**

**ADDRESS: C/28, THE LOWER MALL, PATIALA-147001
WEBSITE: www.psou.ac.in**



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME COORDINATOR :

Dr. Monika Pathak

**Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala**

PROGRAMME CO-COORDINATOR :

Dr. Gaurav Dhiman

**Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala**

COURSE COORDINATOR :

Dr. Karan Sukhija

**Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala**



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

BCA-4-02T: Computer Networks

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 40%

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short q questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Course: Computer Networks	
Course Code: BCA-4-02T	
Course Outcomes (COs)	
After the completion of this course, the students will be able to:	
CO1	Build an understanding of the fundamental concepts of computer networking.
CO2	Identify the different types of network topologies and protocols.
CO3	Understand the concept of networking models, protocols, functionality of each layer
CO4	Identify the function of a firewall, and how it keeps a computer secure and safe from viruses.
CO5	Understand the concept of Firewalls for Network Security.

SECTION-A

Unit I: Basic Concepts: Components of data communication, modes of communication, standards and organizations, Network Classification, Network Topologies; Transmission media, network protocol; layered network architecture.

Unit II: Models: Overview of OSI reference model; TCP/IP protocol suite. Physical Layer: Cabling, Network Interface Card, Transmission Media Devices- Repeater, Hub, Bridge, Switch, Router, Gateway; Transmission impairments.

Unit III: Data Link Layer: Framing techniques; Error Control; Flow Control Protocols; Shared media protocols - CSMA/CD and CSMA/CA.

Unit IV: Network Layer: Virtual Circuits and Datagram approach, IP addressing methods - Sub netting; Routing Algorithms (adaptive and non-adaptive)

SECTION-B

Unit V: Transport Layer: Elements of transport protocols – Addressing, Connection establishment and release, Flow control and buffering, Transport services, Transport Layer protocol of TCP and UDP.

Unit VI: Session and Presentation Layer: Session Layer – Design issues, remote procedure call. Presentation Layer – Design issues, Data compression techniques, Cryptography.

Unit VII: Application Layer: Application layer protocols and services – Domain name system, HTTP, E-mail, WWW, telnet, FTP, SMTP.

Unit VIII: Network Security: Common Terms, Firewalls, Virtual Private Networks

Reference Books:

- B.A. Forouzan, “Data Communication and Networking”, 4th Edition, Tata McGraw Hill, 2017.
- A. S. Tanenbaum, “Computer Networks”, 5th Edition, Pearson, 2011
- D.E. Comer, “Internetworking with TCP/IP”, Vol. I, Prentice Hall of India, 2015
- W. Stalling, Data & Computer Communication, 8th edition, Prentice Hall of India, 2013
- D. Bertsekas, R. Gallager, Data Networks, 2nd edition, Prentice Hall of India. 1992.

UNIT I: INTRODUCTION TO COMPUTER NETWORKS

STRUCTURE

- 1.0 Objectives**
- 1.1 Introduction**
- 1.2 Components of Data Communication System**
- 1.3 Modes of Communication**
 - 1.3.1 Simplex**
 - 1.3.2 Half Duplex**
 - 1.3.3 Full Duplex**
- 1.4 Standards and Organizations**
- 1.5 Network Classification**
 - 1.5.1 Personal Area Network**
 - 1.5.2 Local Area Network**
 - 1.5.3 Metropolitan Area Network**
 - 1.5.4 Wide Area Network**
- 1.6 Network Topologies**
 - 1.6.1 Bus Topology**
 - 1.6.2 Ring Topology**
 - 1.6.3 Star Topology**
 - 1.6.4 Mesh Topology**
 - 1.6.5 Tree Topology**
 - 1.6.6 Hybrid Topology**
- 1.7 Transmission Media**
 - 1.7.1 Guided Media**
 - 1.7.2 Unguided Media**
- 1.8 Network Protocols**
- 1.9 Layered Network Architecture**
- 1.10 Summary**
- 1.11 Review Questions**

1.0 OBJECTIVES

- Understanding the data communication system
- Analyzing the differences among modes of communication
- Identifying the organizations for standardization in data communications
- Understanding the network types and topologies
- Understanding the type of transmission media
- Knowing the concept of protocols

1.1 INTRODUCTION

Data communication is the exchange of data between two devices using any communication media. The main objective of data communication is to enable exchange of data between any two points in the network. For data communication both hardware and software are required. This module provides an introduction to data communication and covers fundamental topics related to the actual transmission of data.

1.2 COMPONENTS OF DATA COMMUNICATION SYSTEM

There are five constituent components of data communication.

- Data
- Sender
- Receiver
- Communication media
- Communication protocols

Data: It is piece of information that may be a text, images, audio or video file.

Sender: It is a source from where communication will start. It can be any device eg. computer, mobile, laptop etc.

Receiver: It is a destination where the data communicated will be received. Like sender, receiver can also be a computer, mobile, laptop etc.

Communication media: It is a medium through which data will be physically communicated from sender to receiver. It can be air, cable or optical fibre.

Communication protocols: It is the set of rules agreed upon by sender and receiver for communicating, in absence of rules the communication cannot happen.

A typical data communication system can be represented in fig. 1, where the data communication has been depicted.

In case of communication among two persons, at any time one is sender, other is receiver and voice or sound is communicated through air. The protocol in this communication is to choose a common language for communication and preferably one should speak i.e. send, at any time and other should listen i.e. receive.

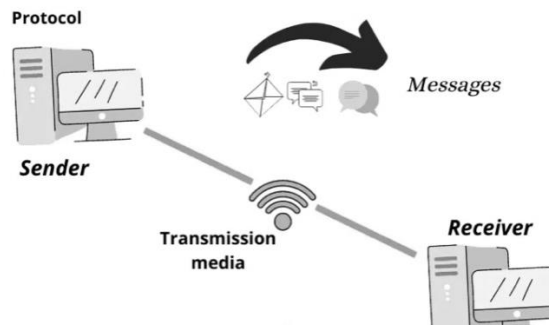


Fig 1: Data Communication System

Similar example in case of email communication will have sender, receiver, data to be emailed, communication through internet, which is wireless/ wired connectivity and the most commonly used protocol - simple mail transfer protocol (SMTP).

1.3 MODES OF COMMUNICATION

The direction of exchange of data between two devices connected through communication media is termed as communication mode. Many such interconnected devices are termed as data communication network.

There are three types of modes of communication:

- Simplex
- Half duplex
- Full duplex

1.3.1 Simplex

This is the simplest mode of communication. The communication in this mode is unidirectional. The sender can only send data and receiver can only receive it.. It is like a one-way traffic, where the vehicles use the full road width to commute and there are no traffic issues. Similarly in mode, the full capacity of communication media channel is used by sender for transmission so it is fast. Radio and television transmission are its examples. This mode suffers from drawback that even the acknowledgement of data received cannot be communicated to sender.

1.3.2 Half Duplex

In this mode communication can be either way between two devices. Both the connected devices can act as sender or as receiver. At any point of time one device will act as sender and other as receiver and vice-versa. This mode also has an advantage of using full capacity of communication media channel. Walkie-Talkie, internet browsers are the examples of this mode of communication.

This mode suffers a drawback that there may be delay in communication as at any given point of time it is still unidirectional.

1.3.3 Full Duplex

The communication in this mode is bidirectional. Both the connected devices can send and receive data simultaneously at any point of time. It is like two-way traffic where vehicles can move in both the directions but the width of road is divided into two paths for accommodating traffic moving in both directions. Mobile phones is example of this mode.

The communication is fast as compared to other two modes but it suffers from a drawback that the communication channel in any direction is half as compared to simplex or half duplex mode.

Amongst the three modes, simplex uses full channel capacity whereas half duplex uses full capacity for communication along one direction at a time and full duplex uses half the capacity for communication in any direction. The bandwidth which is the amount of data communicated between the connected devices in given amount of time is more in full duplex as compared to other two modes. Fig. 2 depicts these three modes of communication.

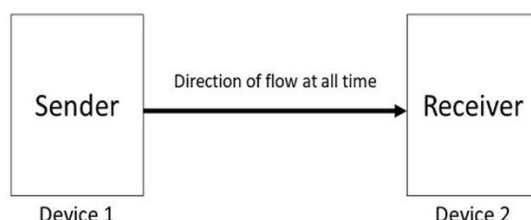


Fig 2a: Simplex Mode

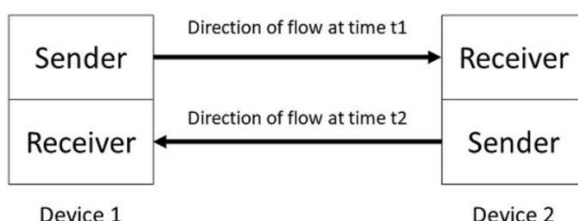


Fig 2b: Half Duplex Mode

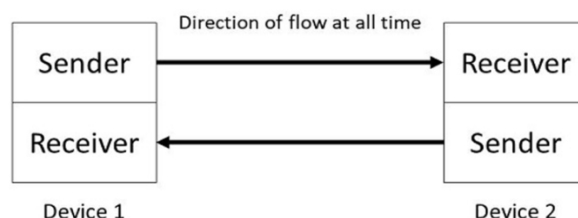


Fig 2c: Full Duplex Mode

1.4 STANDARDS AND ORGANIZATIONS

Standards are required for interoperability at national and international levels. There are organizations that define and maintain the set of rules, guidelines or protocols termed as standards for data communication. These organizations work in association with government

and manufacturers to develop, coordinate and maintain the standards for data communication. The major standards organizations for data communication are:

International Organization for Standardization (ISO): It is a non-governmental, international organization having membership from national standards body. The major functionality of ISO is to develop models and standards ensuring system compatibility, quality and cost effectiveness. Open System Interconnection (OSI), a basic reference model for networks was developed by ISO.

International Telecommunications Union-Telecommunication Sector (ITU-T): The most significant role of ITU-T is to set and regulate the standards for telecommunications worldwide. The major functionalities of ITU-T are standardizing telecommunication technologies, services, tariffs etc. They play a vital role in allocating radio frequency bands for different services. ITU-T defines series of communication standards, most popular among them are V-series that define methodologies for data exchange over public telephone system by using modems etc.

Institute of Electrical and Electronics Engineers (IEEE): It is an international professional body with members from different nations. IEEE standards association (IEEE-SA) is the global standard developer for industry including telecommunications and information technology. IEEE 802 family of network standards is the most commonly adopted by the industry for wireless and wired communications.

American National Standards Institute (ANSI): It is a private organization in United States (US) that manages voluntary standards system in the country. The major role of ANSI is to bring together the various stakeholders i.e. government, industry and users to voluntarily develop and consensus standards. American Standard Code for Information Exchange (ASCII), a character encoding standard widely used for data communication is a contribution of ANSI.

Electronic Industries Association (EIA): It is aligned with ANSI to promote standards for electronic equipment manufacturing for data communication. The physical connection interface standards were defined by EIA. The most commonly used connector Recommended Jack #45 (RJ-45) in networking is the result of standards by EIA.

Internet Engineering Task Force (IETF): It is the leading internet standards organization. With a focus to develop open standards using open processes for enhancing quality of internet. It is large open community of network designers, internet vendors, researchers and users. IETF is responsible for managing internet protocol stack, including Transmission Control Protocol and Internet Protocol (TCP/IP).

Telecommunications Industry Association (TIA): It is an association of global manufacturers and suppliers of hi-tech Information and Communication Technology (ICT) products. It prepares guidelines to produce equipment like cellular towers, telephone terminal equipment, satellites, healthcare ICT, mobile device communication etc.

Internet Architecture Board (IAB): It is a committee of IETF to provide architectural oversight on internet protocols and to give technical direction for internet development and evolution as a global communication platform. The major functions of IAB are to provide foundation for privacy and security in internet, to provide direction to integrate internet with IoT and mobile networks and to promote open internet without special controls.

Internet Research Task Force (IRTF): It focuses on promoting research for evolution of internet by creating research groups for long term research goals. It works in parallel to IETF that focuses on short term goals. Primary task of these groups are to work on topics related to internet protocols, applications and architecture.

Internet Society (ISOC): It manages internet standards and works for policy development. ISOC has mission to ensure open internet development.

Internet Corporation for Assigned Names and Numbers (ICANN): It majorly works for IP address space allocation and top-level domain name system management.

These organizations work to create uniformity across all the stakeholders i.e. manufacturers, consumers and government agencies. Their primary function is doing research, coordinating with other organizations, developing and maintaining technical standards for communication and networks.

1.5 NETWORK CLASSIFICATION

There are different network architectures and protocols with different characteristics and application domains. Networks can be classified according to scale or scope and transmission technology. The networks can be classified based on their scale i.e. the physical size as:

1.5.1 Personal Area Network (PAN)

As the name suggests, it is for a person. It is generally a wireless communication network where personal devices autonomously detect and connect with each other in the range of upto 30 fts. Smartphones, laptops and other wearables devices comes under this category.

1.5.2 Local Area Network (LAN)

It is a small size network covering few buildings or a campus upto few kilometers. It is mainly used to connect workstations or personal computers in offices, factories, institutes for resource sharing. Sharing of printer is a typical example of LAN.

1.5.3 Metropolitan Area Network (MAN)

It is a network covering buildings to cities. It generally covers the geographical area of about 100km and usually interconnects LANs using optical fibre links. A cable television network, fiber distributed data interface (FDDI) and asynchronous transfer mode (ATM) are examples of MAN.

1.5.4 Wide Area Network (WAN)

It covers large geographical area such as country or entire continent for long-distance transmission of data. The communication speed is slow as compared to LAN and MAN. WAN is not owned by any organization like LAN. Internet, 3G and 4G mobile broadband services are examples of WAN.

Fig 3 diagrammatically represents the networks based on physical size.

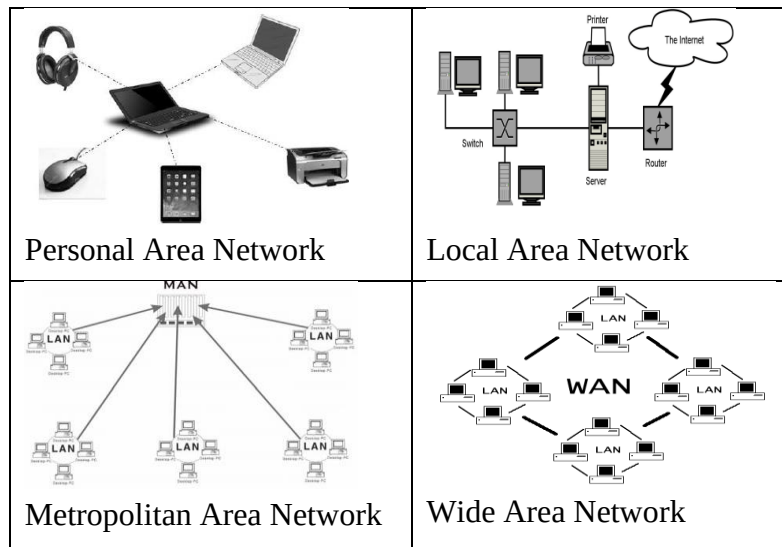


Fig 3: Types of networks based on physical size

1.6 NETWORK TOPOLOGIES

Network topology refers to the arrangement of network comprising of placement of nodes and links and also the physical or logical description of links. The physical description focuses on how the network is actually laid whereas the logical description defines the actual flow of data in the network.

Network topology plays an import role in functionality of the network. Choosing right topology helps in enhancing performance, ease in finding faults, effective resource sharing.

The two main categories of network topology are

- **Point-to-point topology:** A simplest topology where two devices or nodes are connected through a common link. It has the advantages of providing highest bandwidth, simple and fast, low latency and easy to maintain. Personal computer connected to printer is an example of point-to-point topology.
- **Multipoint topology:** In this topology three or more nodes are connected through a single transmission media. Buss or mesh type topologies are example of multipoint topologies.

Multipoint topologies can be categorized as:

1.6.1 Bus Topology

It consists of multiple nodes connected through a single cable with a terminator at each end. One of the nodes act as sender or server and transmits data form one end to another. As the data reaches to each node, it checks the address (MAC or IP address), if it matches with its address, the node processes that data otherwise it just passes the data. Once the data reaches to other extreme, the terminator removes the data from the line. The main line or cable acts as the backbone for the complete network in this topology. It is also known as multi-drop, linear or horizontal bus and is mainly suited for small networks like LAN. Fig 4 represents the bus topology.

Advantages:

- Simple to install and easy to add nodes

- In case of failure of node, other nodes will not be affected
- Less cabling cost as compared to other topologies
- Cost efficient

Disadvantages:

- Strength of signal decreases when nodes are more, leading to low efficiency
- The entire network fails if the backbone cable is damaged
- Congestion and traffic on bus as there is single communication media connecting nodes
- Due to limited bus length, limited number of nodes can be connected
- Broadcasting of message leads to security issues and risks

1.6.2 Ring Topology

In this topology, each node is connected to two neighbouring nodes on either side and form the shape of ring. The message by the sender travels through the ring in the same direction either clockwise or anticlockwise, till it reaches to receiver.

The data flow in this topology is based on token passing, only the node having token can transmit. The receiver takes data from token and sends back it to sender as an acknowledgement. The token is regenerated and is passed in the ring again.

Advantages:

- Chance of packet collision is minimal
- Transmission speed is high due to Unidirectional data
- Easy installation, robust and easy maintenance
- Less costly as compared to other topologies eg. mesh, hybrid, tree

Disadvantages:

- Failure of any load will lead to network failure
- Data packet has to pass through all the nodes causing security issues
- Addition and removal of any node is difficult

1.6.3 Mesh Topology

It's a kind of point to point topology where each node is connected to every other node.

The total number of simplex links in a mesh topology is $n(n-1)$ and for full duplex it will be $n(n-1)/2$; where n is total number of nodes in the topology.

Mesh can be full mesh or partial mesh depending on all the nodes of partial nodes are connected with each other.

Advantages:

- Dedicated links to facilitate one to one communication
- No congestion on the channel
- In case of failure of single node the communication can be routed through other nodes
- Privacy and security is maintained

Disadvantages:

- Cost of cabling is high

- Installation is difficult

1.6.4 Star Topology

It is point to point connection topology where every node is connected to the centralized hub. Nodes are not directly connected to each other like in mesh. Sender sends the data to hub, which further sends it to receiver. The hub finds the addresses of all the receiver node and communicates the data.

Advantages:

- Easy to maintain and troubleshoot
- Failure in cable will affect only a single node
- Less expensive, easy to add new nodes

Disadvantages:

- Failure of central hub will lead to network failure
- Central hub has limitation to handle the nodes due to limited input-output ports

1.6.5 Tree Topology

In this topology nodes are connected directly or indirectly to main bus cable like branches of a tree. It is a combination of star and bus topologies where number of star networks are connected using bus.

Advantages:

- Large distance networks can be covered
- Failure of any node will not affect any other nodes
- It is scalable topology with easy in fault finding

Disadvantages:

- More cabling required as compared to star and bus topologies
- Configuration is complex
- The failure of main bus will lead to failure of complete network

1.6.6 Hybrid Topology

It is a combination of two or more topologies. It can be combination of any of the five topologies discussed above. It can be star-ring or star bus topologies.

Advantages:

- It has the advantages of multiple topologies being used
- It is flexible, reliable and scalable

Disadvantages:

- Design is complex
- It is expensive as compared to other topologies

Fig. 4 represents the type of network topologies commonly used in data communications.

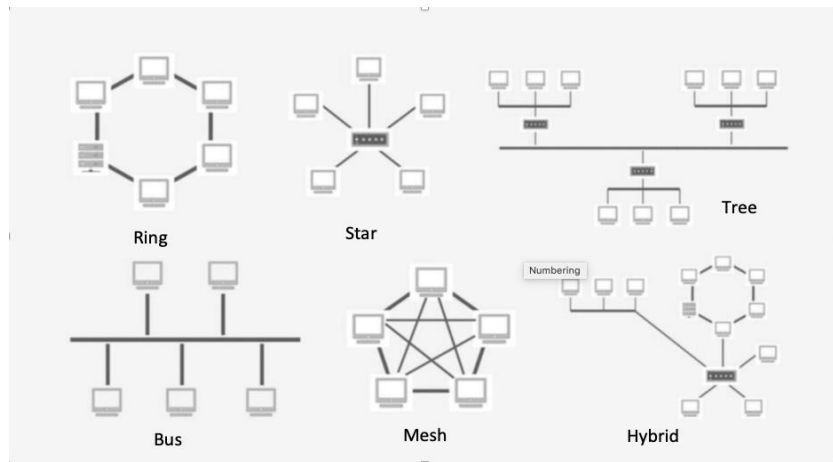


Fig 4: Types of network topologies

1.7 TRANSMISSION MEDIA

A physical path or channel between sender and receiver for transmitting data is a transmission media. Data can be in the form of electromagnetic signals or light pulses depending on the type of media.

Transmission media can be categorized into guided and unguided media.

1.7.1 Guided Media

It is a wired or bounded media. The signals are transmitted through a defined path by using physical link. This type of media is secure, provides high speed. It is used for small distances. There are three kinds of guided media used for communication.

- **Twisted pair cable:** The two insulated conductor wires are wound about each other to make twisted pair. Several such pairs are bundled together in a protective sheath. It is most commonly used transmission media. It is of two types:
 - **Unshielded Twisted Pair (UTP):** During transmission, this cable can block the external interference without using protective shield. It is majorly used for telephonic applications, LAN networks. It is used for short distance transmission due to attenuation. It is easy to install, comparatively cheap and has high speed.
 - **Shielded Twisted Pair (STP):** The cable consists of protective shield to block external interference. It is used where fast data rate is required. Its performance is better than UTP as it eliminates cross talk and provides better communication speed. STP is more expensive and bulky than UTP. It is mainly used in the network with high electro-magnetic interference. IBM token ring topology uses STP.
- **Coaxial Cable:** It is commonly termed as “coax”. It consists of inner conductor surrounded by concentric conducting shield with insulator in-between the two. Coax has outer jacket of insulating material. Its major application is in cable TV network. It is less expensive as compared to twisted pair cable. It is easy to install and provides

high bandwidth which is the maximum amount of data transmitted in given amount of time. Any failure in the cable disrupts the entire network.

- **Optical Fibre Cable:** Optical fibre is a thin strand of glass acting as a wave guide for transmitting light signals over long distance. It works on the principal of total internal reflection. It has two layers; core and cladding surrounding core. Cladding helps in attaining total internal reflection of light in core. Based on the size of core, it can be categorized into single-mode or multi-mode. It is used in telecommunications, medical use etc. It provides very high bandwidth, fast speed and negligible interference. It is light weight. Handling of optical fibres is difficult as it is very fragile. Its cost is high as compared to other two cables.

Fig. 5 shows the guided transmission media i.e. two types of twisted cables, UTP & STP, coaxial cable and optical fibre

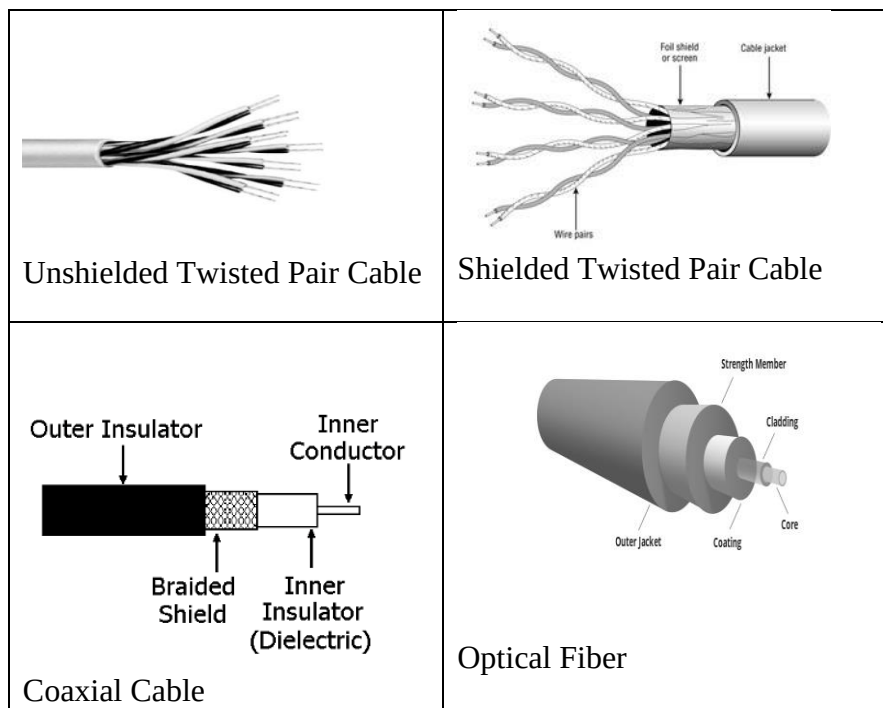


Fig 5: Types of guided transmission media

1.7.2 Unguided Media

It is wireless or unbounded media. There is no physical media for transmitting data but air. Antennas are used to send and receive data. It is generally used for covering large distances. Data is not secure while transmitting through this media. It is of three types.

- **Radio Transmission:** The radio waves used for this type of transmission are low frequency electro-magnetic waves ranging from 3KHz to 1GHz. It can penetrate walls so the signal can be received in-side the building as well. It uses a sky propagation thus the signal can be broadcasted over large distance. It is used for AM and FM

radios. Simultaneous data transmission is a problem in radio waves due to its low frequency.

- **Microwave Transmission:** These waves have frequency ranging from 1 to 300GHz. These waves are unidirectional so it propagates in line-of sight mode. The sender and receiver antennas should be aligned for transmission and reception. They are used for mobile phone communication or for television distribution. It supports large bandwidth so large amount of data can be transmitted. The equipment cost and the installation cost is very high.

It can be categorized into Terrestrial microwave transmission, where both sender and receiver antenna are on ground and Satellite microwave transmission, where sender antenna signal is received by satellite, amplified and transmitted to receiver antenna on ground. Terrestrial is used for telecommunication and satellite is used for global positioning system.

- **Infrared Transmission:** It has a frequency range from 300GHz to 400GHz. It is used for short range communication. It cannot penetrate the walls so the interference is less. It is mainly used for remote operations of home appliances and devices.

1.8 NETWORK PROTOCOLS

Set of rules determining how the data will be communicated between two nodes connected in a network. With the well-defined protocols the nodes can communicate with each other irrespective of differences in their structure, design or internal processes. Protocols are designed as per industry standard by the various organizations mentioned in the earlier section.

The major functionality of network protocols is to provide:

- means to identify the nodes in the network and to make wired or wireless connection for communication
- formatting rules for packaging, sending and receiving the data
- process for establishing and terminating connection
- acknowledgement services wherever required
- data compression and encryption features for faster and secure communication

Based on the primary actions performed by protocols they can be broadly categorized as:

- **Communication Protocols:** They allow the nodes or devices to communicate, transfer files or accessing the internet.
- **Network Management Protocols:** They specify the procedures for the optimal performance of the network.
- **Security Protocols:** They ensure the protection of data communicated through channels is protected from unauthorized users.

There are few common protocols which we come across on daily basis, hypertext transfer protocol (HTTP), short message service (SMS), simple mail transfer protocol (SMTP), internet message access protocol (IMAP) are few among those.

Protocols are designed based on a layered architecture of the networks such as the OSI reference model or TCP/IP model.

1.9 LAYERED NETWORK ARCHITECTURE

The whole process of data communication is divided into tasks whereas data communication network is divided into layers. Each layer handles the specified tasks. The purpose of layered architecture of network is to reduce the design complexity. Network is organized as layers build upon one another. Each layer get the services of layers below it and provides services to layers above it.

Advantages:

- It reduces the design complexity of network
- Due to the modular approach of the layered architecture, the changes made in one layer will not affect the other layers
- It provides the abstraction of logical communication with corresponding layer in another node. Logically nodes can communicate at different layers with another node.

The layered architecture has several advantages but layering add to overheads in communication.

There are three basic elements of layered network architecture :

- Service: set of actions that a layer offers to higher layers. Service can be accessed through service access point (SAP) by sending service data unit (SDU).
- Protocols: set of rules to exchange information among layers. It is done by exchanging protocol data unit (PDU).
- Interfaces: to communicate the data or message to another layer. Interface data unit (IDU) is shared among layers.

The two models in the networking based on the layered architecture are Open Source Interconnection (OSI) model and Transmission Control Protocol/ Internet Protocol (TCP/IP) model. OSI model is a generic model based on the functionality of layers while the TCP/IP model layout the standards on which internet was developed.

1.10 SUMMARY

This module covers the basics of data communications system, its components, modes of data communication i.e. simplex, half duplex and full duplex. The organizations that define standards for data communication for the global network or internet have been detailed. Various network topologies that are being used while setting the network have been discussed along with their advantages and disadvantages. In general hybrid topology is the most preferred that has the advantages of multiple topologies. According to the size and scope of the network, its categorization in PAN, LAN, MAN and WAN have been detailed. Transmission media plays a very major role in data communication. Characteristics of media decides the bandwidth it can manage also the reliability with which it can communicate data. Both the guided and unguided media have their own utility in wired or wireless types of networks. Another important part in the data communication is the protocols following which the sender can communicate with receiver.

Broad categories of the protocols have been discussed in the chapter. Also, the concept of layering in the networks have been introduced in this module.

1.11 REVIEW QUESTIONS

- a. List the five major components of data communication system.
- b. Differentiate between half duplex and full duplex transmission mode. Give one example each for these modes.
- c. What is the need for defining standards for data communication? List any three organizations that design the standards.
- d. According to the physical size, how can we classify networks? The most commonly used internet will come under which specific category.
- e. Differentiate between point to point and multipoint topologies.
- f. Which topologies will result in network break down if only the single device goes down?
- g. What are the advantages of mesh topology over ring topology?
- h. Which topology is practically used in setting up a network? Specify the reasons for your answer.
- i. Compare the types of guided transmission media. Which one would be preferred for transmitting data to a large distance?
- j. Why the infrared waves used in the remotes of home devices do not interfere with each other?
- k. Enumerate the importance of protocols in data communication.

REFERENCES & FURTHER READING

- Data Communication & Networking by BehrouzForouzan
- Computer Networks by Andrew Tannenbaum
- Data and computer Communications by William Stallings
- Computer Networking: A Top-Down Approach by James Kurose and K.W. Ross

UNIT 2: MODELS AND TRANSMISSION MEDIA

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Network Architecture

2.3 Concept of Layered Network Architecture

2.4 Open Source Interconnection (OSI) Reference Model

2.4.1 Introduction to OSI Model

2.4.2 Layered Architecture of OSI Model

2.4.3 Description of Layers in the OSI Model

2.5 TCP/IP Model

2.5.1 Introduction to TCP/IP Model & its layers

2.5.2 OSI and TCP/IP Model Comparison

2.5.3 Description of Layers in the TCP/IP Model

2.5.4 Addressing in TCP/IP

2.6 Physical Layer

2.6.1 Cabling

2.6.2 Comparison of Optical Fiber and Copper Wire

2.6.3 Network Interface Card

2.6.4 Transmission Media Devices

2.7 Transmission Impairments

2.8 Summary

2.9 Review Questions

2.10 References & Further Reading

2.0 OBJECTIVES

- a. Understanding the concept and importance of layered network architecture
- b. Understanding the OSI and TCP/IP model
- c. Difference between OSI and TCP/IP model
- d. Understanding the functioning of physical layer
- e. Get introduced to commonly used devices in networks
- f. Understanding impairments in transmission of data

2.1 INTRODUCTION

Data Communication Network has basic underlying reference model known as Open System Interconnection (OSI) model and practically working model which is TCP/IP model. The physical layer common in both of these models has specific functionality including the functioning of network devices for transmission of data. This module gives an understanding of OSI and TCP/IP model and covers the introduction of physical layer focusing on devices and flaws in transmission.

2.2 NETWORK ARCHITECTURE

Network architecture is the design of a computer network. It is a framework for the specification of a network's physical components and their functional organization and configuration, its operational principles and procedures, as well as communication protocols used.

2.3 CONCEPT OF LAYERED ARCHITECTURE

The major task of the network is to transfer data from sender node to the receiver node. This major task can be divided into well-defined smaller sub-tasks.

Each of this sub-task consist of various processes. Each sub-task will accept the output from previous sub-task as input for its processes and will provide its output as an input to next sub-task. Each of these sub-tasks is performed by a separate layer in the network architecture. Hence the name layered architecture. In general the complex tasks are assigned to higher layers and simpler ones are assigned to middle or the lower layers.

2.4 OPEN SOURCE INTERCONNECTION (OSI) REFERENCE MODEL

2.4.1 Introduction to OSI Model

OSI model was developed by International Organization for Standardization (ISO) as a reference model for computer networks. The objective to have a reference model was to allow different platforms which is a combination of hardware, software and operating system to communicate with each other. The protocols are well defined for each layer in OSI reference model.

The sub-tasks in OSI model are divided into seven categories, each one is taken up by each of the seven layers in the model. These layers from lower to higher are listed below:

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer

5. Session Layer
6. Presentation Layer
7. Application Layer

The functionality of each of these layers is explained in next section.

2.4.2 Layered Architecture of OSI Model

The OSI model has seven layers with well-defined tasks. The layered architecture of the model has following features:

- Message or data from sender node to the receiver node pass through all the layers from top to bottom i.e. application to physical layer and then it is delivered to receiver through the wired or wireless transmission media.
- Message or data received by receiver node passes from physical layer to the application layer.
- While transmitting the data, the sender and receiver may not have point-to-point connection i.e. the data may have to pass from intermediary nodes to reach to the receiver node. In that case intermediary nodes only involve the first three layers i.e. physical, data link and network layer for passing the data to next node.
- The data link layer decides about the next intermediary node in the communication path whereas the network layer identifies the end receiver.

Fig. 1 depicts the data flow from sender to receiver through various layers of OSI model showing the intermediary nodes between sender and receiver for transmitting the data.

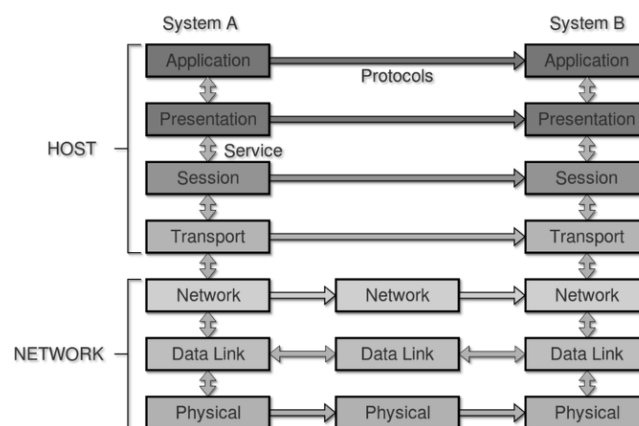


Fig 1: Open System Interconnection (OSI) model

- For the communication, at sender node, each layer add its information known as header, to the data or message received from the layer above. After adding the header information layer passes the message to the layer below it. This is known as data encapsulation and is represented in fig. 2 below.

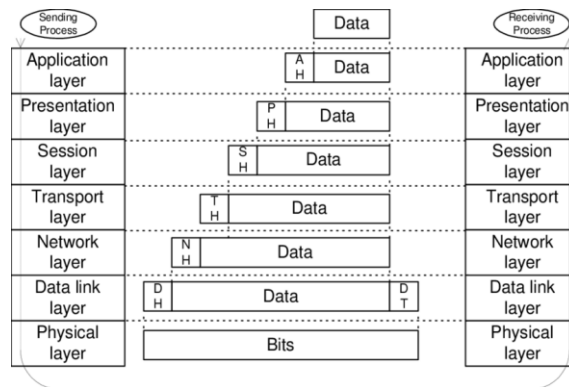


Fig 2: Header information at layers of OSI model

- At receiver node, the reverse process happens, each layer removes the information pertaining to the specific layer from the message received from lower layer and pass the message to the layer above.
- The function at each layer is specific and different from that of other layers.
- At sender, each layer calls for the service offered by layer below it but at receiver each layer call for the service offered by layer above it.
- The abstraction at the layer level provides the communication between the respective layers at sender and receiver nodes.
- The message among layers is communicated through the interface which defines the service that must be provided by that layer.

Fig 3., depicts the layered architecture with interfaces between layers.

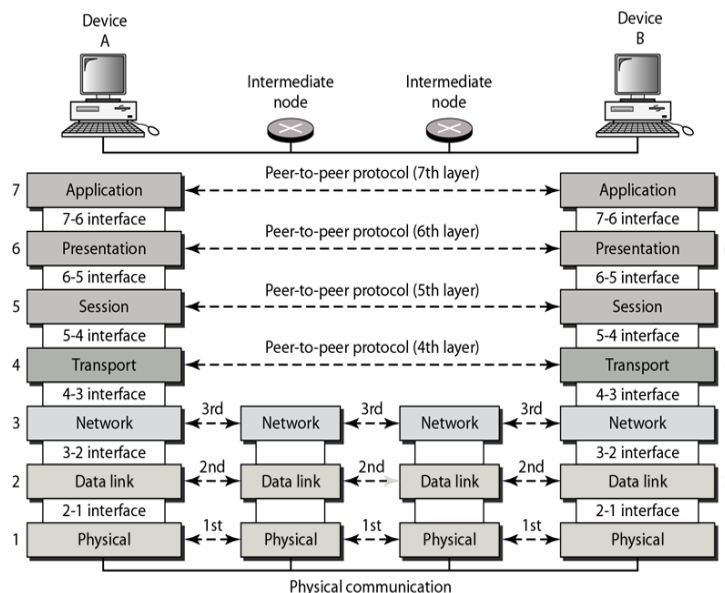


Fig 3: Interfaces and abstraction at layers in OSI model

2.4.3 Description of Layers in the OSI Model

The functions of each layer in the OSI model can be described as:

1. **Physical Layer:** The function of physical layer is to transmit the data through transmission media from sender to receiver.
 - a. A standard interface is provided by the layer to transmission media. This interface defines type of transmission media, its specifications and bit by bit delivery of message.
 - b. At sender, physical layer receives encapsulated message from data link layer and encodes the message into signal according to the transmission media.
 - c. At receiver side, the signal is received by receiver through the transmission media, after decoding, signal is sent to data link layer. Fig 4. Depicts the physical communication of data from sender to receiver.

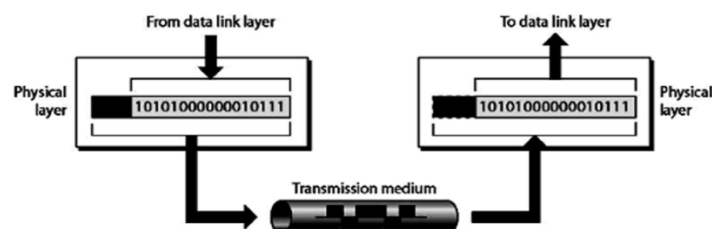


Fig. 4: Transmission to and from Physical layer

- d. Physical layer converts the data from combination of 1's and 0's into signals from communication at sender and vice-versa at receiver.
- e. The data transmission rate i.e. bits/ sec is decided and managed by physical layer.
- f. The synchronization between sender and receiver is maintained by this layer. The same bit rate for both the nodes is pre-decided and data is transmitted in accordance to synchronized clocks.
- g. The type of connection i.e. point-to-point or multi-point and accordingly the type of topology discussed in previous module, is defined by physical layer.
- h. The transmission mode i.e. simplex, half-duplex or full duplex between sender and receiver is defined by this layer.

In totality, all the specifications and processes related to actual transmission of data from sender to receiver is defined by the first layer i.e. physical layer in OSI model.

2. **Data Link Layer:** The reliability in communication of data is provided by data link layer through error detection and correction process.
 - a. Data link layer on the sender side receives message from network layer and divides into smaller fixed size units termed as **frames**, which is further sent to physical layer for communication as a bit stream.

- b. At receiver side this layer receives bit stream of message from physical layer and does the re-grouping to form a frame for sharing with the network layer. This process of re-grouping the bit stream into frames is known as **framing**.
 - c. The header information added by data link layer in frame includes the physical address of the sender and receiver in case the sender and receiver are on the same physical network otherwise instead of receiver it stores the physical address of the intermediary node to whom data would be communicated next.
 - d. Data link layer manages the flow control mechanism so as to avoid any loss or overflow of data due to speed mismatch of sender and receiver.
 - e. This layer offers error control by identifying lost or damaged frames and resending the duplicate frames.
 - f. This layer ensures hop-to-hop transmission of frames from sender to receiver.
- 3. Network Layer:** The main function of network layer is to deliver data packets from sender to receiver across multiple hops.
- a. At sender side, network layer receives message from transport layer and divides it into packets.
 - b. At receiver side the network layer receives frames from data link layer and converts it into packets and communicates to transport layer.
 - c. To deliver the data from sender to receiver, the data is routed through various intermediary nodes.
 - d. To recognize the devices in the network, this layer uses internet protocol (IP) addresses, which is a unique logical address for the nodes connected in network.
 - e. Network layer's main function is routing, the packets are numbered and transmitted from sender to receiver through multiple paths based on congestion, number of hops etc. At receiver the numbered packets are combined in a sequence to receive the full message.
 - f. During the packet transmission the intermediate nodes check about the best possible path for next hop.
- 4. Transport Layer:** The message transmitted from sender to receiver may be for a specific process whereas multiple processes are running on sender and receiver nodes. This layer ensures the data to be delivered to the correct process at the receiver node and hence achieve process to process delivery.
- a. At sender the transport layer receives data from session layer and divides it into segments which are numbered and delivers it to the network layer.
 - b. At receiver side the packets received from network layer are combined to make a segment to be further delivered to session layer.
 - c. Every process is assigned a port number on the node, transport layer checks the port number at the receiver node to ensure the delivery to correct process.
 - d. Transport layer provides connection oriented and connection less services for communication.
 - e. Like data link layer, transport layer also provides flow control and error control but for end to end delivery.

5. **Session Layer:** This layer handles a task for establishing and synchronizing session generally termed as dialog between sender and receiver.
 - a. At sender side, it accepts data from presentation layer add checkpoints, synchronization bits and delivers data to transport layer.
 - b. At receiver, session layer receives data from transport layer removes checkpoints and delivers data to presentation layer.
 - c. Checkpoints ensure that the data before checkpoint has been successfully received and will not be re-transmitted.
6. **Presentation Layer:** This layer helps overcoming the challenge of data transmission to devices on different platforms. It performs translation, encryption and data compression.
 - a. At sender side it receives the data from application layer adds header containing information related to compression and encryption and delivers it to presentation layer.
 - b. At receiver side it receives data from session layer, decompresses and decrypts the data as per the encoding scheme agreed upon by both sender and receiver and delivers it to application layer.
7. **Application Layer:** This layer provides the user interface to communicate data from sender to receiver by providing specific services, eg. email transfer which is known as X400 service, file transfer known as FTAM service etc.

The major functions of these seven layers can be summarized as:

Physical Layer: Transmitting data as a bit stream from one to another hop and providing specification of communication media and type of communication.

Data Link Layer: Hop to Hop delivery of frames to receiver

Network Layer: Delivery of packets from sender to receiver by deciding best routing path.

Transport Layer: Providing process to process delivery of message.

Session Layer: Establishing, managing and terminating the sessions.

Presentation Layer: Providing translation, encryption and compression for disparity in platforms of sender and receiver.

Application Layer: Providing the access of network to the user.

2.5 TCP/IP Model

2.5.1 Introduction to TCP/IP Model & its layers

The TCP/IP model is the practical model, that is currently being used in networks and internet. It was developed by Department of Defence, United States. This model is a collection of protocols termed as protocol suite. Based on the two major protocols i.e. Transmission Control Protocol (TCP) and Internet Protocol (IP), this model got its name as TCP/IP model. There are no standard specifications followed universally for describing TCP/IP model but it has fewer layers as compared to OSI model. For practical purpose this model divides the tasks in four layers, termed from lower to higher layer as:

1. Physical and Data Link Layer

2. Network Layer
3. Transport Layer
4. Application Layer

2.5.2 OSI and TCP/IP Model Comparison

Though the structure of OSI model and TCP/IP model is very similar but there are few differences in the two

- OSI model has seven layers but TCP/IP models has four layered architecture.
- The first layer of TCP/IP model performs the functionality as done by physical layer and data link layer of OSI model.
- The network layer and transport layers of TCP/IP performs the similar functionality as network layer and transport layers of OSI model.
- Application layer in TCP/IP models performs the functionality of application as well as session and presentation layers of OSI model.

2.5.3 Description of Layers in the TCP/IP Model

The functions performed by each layer in TCP/IP model is described below:

1. **Physical and Data Link Layer:** It is combinations of physical and data link layer of OSI model, all the functions performed by these two layers in OSI model is taken up by layer 1 of TCP/IP model. This layer supports all the standard protocols defined for these two layers.
2. **Network Layer:** This layer is also known as internetwork layer. The major protocol defined for this layer is IP. This layer is responsible for source to destination transmission of data.
 - a. IP protocol is a connection-less and unreliable protocol. There is no error checking in this protocol. It simply sends the data and rely on the layers below to get the data transmitted to receiver node.
 - b. The data is divided into same sized packets known as datagrams. Each datagram is numbered and is transmitted through different routes. At receiver side packets or datagrams are received out of order.
 - c. No single connection is established between sender and receiver, and the packets are delivered through best possible route, so the services provided by network layer are connection-less services.
 - d. There is no guarantee in IP for transmission of all the packets. During transmission some packets may get dropped due to congestion or other reasons. This makes IP an unreliable protocol.
 - e. IP provides the maximum efficiency in data delivery as no error detection and error correction services are provided by this protocol, making the delivery fast.
 - f. IP protocols takes the services of four protocols, ARP, RARP, ICMP and IGMP.

- **Address Resolution Protocol (ARP):** This protocol is used to resolve the physical address of the device in the network when the logical address of the node is known.
The physical address or media access control (MAC) address is imprinted on the network interface card (NIC) or LAN card. Logical address is the IP address that uniquely identifies the node in the internet.
 - **Reverse Address Resolution Protocol (RARP):** This protocol helps in identifying the logical or IP address when physical address is known.
 - **Internet Control Message Protocol (ICMP):** This protocol provides the signalling mechanism. The intermediate devices eg. gateway sends the message to sender node if the datagram received by it from sender gets corrupted during transmission.
 - **Internet Group Message Protocol:** This protocol has mechanism to deliver the message of corrupted datagram to a particular group of recipients.
3. **Transport Layer:** The protocols at this layer are responsible for process to process communication of data, similar to the transport layer of OSI model. The three major protocols in this layer are TCP, UDP and SCTP.
- **Transmission Control Protocol (TCP):** It is a connection-oriented protocol, thus a reliable protocol. Before the actual transmission starts, it establishes the connection between sender and receiver.
The data received from application layer is divided into segments and each segment is numbered.
 - **User Datagram Protocol (UDP):** Contrary to TCP protocol, it is connection-less and unreliable protocol. It does not provide flow control and error control mechanism.
 - **Stream Control Transmission Protocol (SCTP):** It is a relatively new protocol that combines the functionality of TCP and UDP protocols. Its major feature is to provide multi-homing, which is multiple connection paths establishment between sender and receiver nodes.
4. **Application Layer:** This layer in TCP/IP model is providing functionality of Session, presentation and application layers of OSI model. The high level protocols eg. file transfer protocol (FTP), virtual terminal protocol (TELNET), domain name service (DNS) are defined in this layer.

2.5.4 Addressing in TCP/IP

There are four different types of addresses defined in TCP/IP model and protocols are defined to resolve the addressing. These addresses are referred to as:

- **Physical Address:** It is the lowest level of addressing and is local to the network to which the node is connected. This address is unique. Media access control (MAC) address is 48 bits physical address printed on the network interface card (NIC) of the node. The size of the address may vary according to the type of network. In ethernet, MAC address is 48 bits (6 bytes).

- **Logical Address:** This logical address of IP address is used for uniquely identifying the node in the internet. The physical address is local to the network so while multiple networks are connected, there are chances of duplication of address. For sender to receiver communication, which are on different networks, physical address is not adequate to uniquely identify the nodes.

All the devices in the internet are uniquely identified by IP address. Two versions of IP addresses are commonly used to identify devices:

IPv4: It's a 32 bit logical address. It can identify maximum of 2^{32} unique devices in the internet. As the number of devices on internet is growing, IPv4 may not be able to allocate unique addresses to the devices.

IPv6: It's a 128 bit address and can support 2^{128} unique devices connected to internet.

- **Port Address:** The logical address helps in identifying the sender and receiver nodes connected to the internet but there may be multiple processes running on these sender and receiver nodes. Google meet, email and other processes are running on the devices, identifying the process for which the data is sent or received, is important. Port address uniquely identifies the process on the devices for which data is transmitted. It is a 16 bit address given to the process. The universal processes have been assigned same port numbers on all the devices, eg. http uses port address 80, telnet uses port address 23, smtp has port address of 25.
- **Application Specific Address:** For the same process there may be multiple windows or tabs open for eg. multiple tabs on web browser. These multiple tabs for same process cannot be uniquely identified by port address, so to identify the unique instances of such processes, application specific address has been defined. Such addresses are specific to user so these are user friendly addresses. Eg. e-mail addresses (abc@gmail.com) and Universal Resource Locator (URL) (www.sggswu.edu.in) are user friendly specific addresses.

TCP/IP model is the practical model that has been evolved for internet.

2.6 PHYSICAL LAYER

Physical layer plays a major role in interaction of actual hardware and the signalling mechanism. The hardware equipment, cabling, transmission frequency, synchronization, pulses representing binary signal are all defined by the physical layer. This layer receives the data from data link layer in the form of frames, converts into a sequence of bits, transmits it to physical layer of receiver or of the intermediary node.

Information can be transmitted through cables in the form of electromagnetic signals. Data to be transmitted can either be digital or analog Digital signals like in file transfer can be represented as voltage pulses which is discrete in nature. Analog signals like voice is represented as a continuous electromagnetic waves.

The few terminologies are important for understanding the functioning of physical layer.

- **Channel Capacity:** The data transmission speed is termed as channel capacity. It depends on bandwidth, encoding and error rate.
Bandwidth is limited by the transmission media, encoding depends on the levels used for signalling and error rate depends on the noise in the channel.
- **Multiplexing:** It is a technique to transmit multiple data streams meant for different receiving nodes over a single medium. Multiplexer is used to combine the data streams whereas de-multiplexer is required at receiver's end to separate out the information.
- **Switching:** It is a mechanism by which data is transmitted from sender to receiver that are not directly connected. In this case the data is transmitted to intermediary devices and forwarded to the next device closest to the receiver. The three different types of switching is circuit switching, message switching and packet switching.
 - **Circuit Switching:** In this technique, a dedicated link is established between the sender and receiver that includes fixed intermediary nodes eg. telephone network.
 - **Message Switching:** A dedicated link in circuit switching will be underutilized if no data is there for transmission between sender and receiver. To overcome this drawback, message- a logical unit of data was communicated through store and forward technique by dynamically deciding the route for communication.
 - **Packet Switching:** It is a type of message switching in which the data is divided in fixed size packets. It is also based on store and forward technique eg. internet.
- **Transmission modes:** The data in the form of 0s and 1s can be transmitted from sender to receiver in serial or parallel mode.
 - **Serial Transmission:** In this transmission bits are sent in a queue one after the other. Single communication channel is required for such mode.
It can be synchronous or asynchronous in nature. Synchronous mode is timing based transmission whereas in asynchronous bit pattern itself identifies the start and end of the transmission.
 - **Parallel Transmission:** Multiple bits are transmitted at the same time through multiple data lines. It is a high speed communication as compared to serial transmission.

2.6.1 Cabling

In the previous module, we discussed about two types of transmission media, guided and unguided. Guided media consists of twisted pair, coaxial and optical fiber cables. We will discuss the types of cables used for networking in this section.

- **Twisted Pair:** It is one of the oldest and the most common type of cable. Twisted pair of cables help in reducing noise by forming antenna by twisting the two cables.

It can be used for transmitting either analog or digital data. These cables are commonly used due to its optimal performance and low cost.

For LAN, type of coaxial cable termed as category 5 or Cat 5 is used. It consists of two insulated wires gently twisted together. Cat 5 has four such twisted pairs put together by insulation. The cables are categorized based on number of twists per meter. Cat 5 cables can be used for speed of 100mega bits per second (Mbps),

cat 5e can be used for speed of 1000Mbps etc. Fig. 5 shows the different categories of coaxial cables.

Cat 6 type of cables can support transmission at rate of 10 giga bits per second (Gbps). All these categories are of unshielded twisted pair (UTP) cables. Cables of Cat 7 category comes under shielded twisted pair (STP) cables.

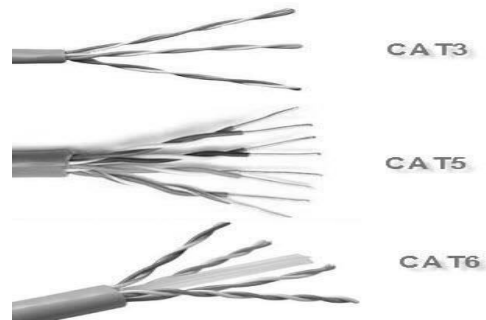


Fig 5: Coaxial cables : cat 3, cat 5 and cat 6

- Optical fiber:** These are used for long-distance networking backbones with the high speed of transmission. The three basic components of optical fiber are : source of light, transmission media and the light detector. The transmission media is ultra-thin fiber of pure glass. A light source is attached to one end of the optical fiber send the light signals through transmission media. The source can be light emitting diode (LED) or semiconductor laser. A light pulse is considered as 1 bit and absence of light is considered as 0 bit. On the receiver end of the fiber, a detector, which is a photodiode, on receiving light pulse converts it to electrical signal.

The transmission through optical fiber follows the physics principle of total internal reflection as shown in figure below.

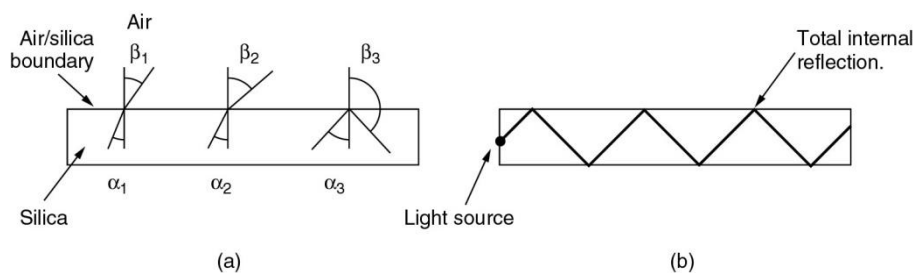


Fig 6: a) Refraction at boundary of two materials b) Total internal reflection

Fiber Cables are similar to coax, excluding the braid. The figure shows the single fiber and view of the sheath with three fibers. Cladding shown in the figure is made up of glass but with lower index of refraction than the core. It helps in keeping the light only in the core.

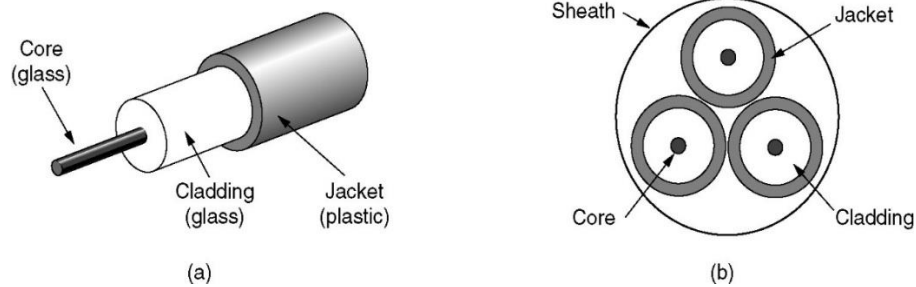


Fig 7: a) Optical fibre b) sheath with three optical fibers

The optical fiber can support high data rate of transmission but is limited to 100Gbps due to response time of the detector to convert light to electrical signal. The transmission through optical fiber suffer from thermal noise.

2.6.2 Comparison of Optical fiber and Copper Wire

Both the transmission medias are being used for communication, but it is important to compare these two medias. The table below gives the comparison of the two based on important parameters:

Parameter	Optical Fiber	Copper Wire
Data transmission	Data is transmitted in the form of light pulses.	Data is transmitted in the form of electrical signals.
Bandwidth	Offers high bandwidth as light communicates faster than electrical pulses.	Low bandwidth in comparison to optical fiber.
Attenuation	The loss in signal strength i.e. attenuation is low.	Attenuation is comparatively high.
Interference	Resistant to electromagnetic interference or lightening.	Vulnerable to electromagnetic interference and lightening.
Durability	High tensile strength, so it is durable.	It is less durable as compared to copper wire.
Installation cost	High	Low
Security	Signals cannot be tapped, hence it is secure.	It can be easily tapped.

The optical fibre has many advantages over copper wire, hence it is replacing the twisted pair cables very fast.

2.6.3 Network Interface Card (NIC)

NIC is a hardware component and is inbuilt in the computer. It connects the computer with other devices on the network. It is also known as network or ethernet adapter, LAN card, network controller etc.

Functions of NIC

- It converts data into digital signal like a translator.
- It offers both wired and wireless communications.
- NIC operates on layer 1 i.e. physical and layer 2 i.e. data link layer of OSI model.

Components of NIC

The main components of NIC are as follows:

- External Memory: It is used to store data temporarily and uses the data while processing the communication.
- Connectors: They are used to make the physical link between cables and ethernet port.
- Processor: It converts the data into signal format for communication.
- Jumpers: They are used to control the communication by turning on or turning off the switch.
- Routers: It is a NIC device to provide wireless connectivity.
- MAC address: It is unique address given to NIC on which the message is communicated. It is also known as physical address.

Types of NIC

NIC are of two types: Ethernet and Wireless Networks.

Ethernet NIC: It is a slot for the ethernet cable. The other end of the cable is connected to modem. It is most widely used in LAN, MAN and WAN. 5-Base T, 10-Base T, 100-Base T and gigabit Ethernet are commonly used NICs.

Wireless NIC: It consists of small antenna integrated onto NIC that helps in communication among devices which are connected wirelessly. Fiber data digital interface (FDDI) is an example of wireless NIC.

NIC has certain advantages:

- It provides reliable connection.
- It allows multiple devices to be connected using different ports of NIC.
- It facilitates bulk data sharing.

NIC has the following disadvantages:

- Wired NIC is not portable as compared to wireless NIC
- Proper configured is required for getting better communication.
- It does not provide data security.

NIC is used for data exchange over the internet. It is applicable for devices like bridges and repeaters and also for hubs, switches and routers in case of wired communication device.

2.6.4 Transmission Media Devices

The common transmission media devices used in networks are as follows:

- **Repeater:** This device operates at physical layer. Its function is to regenerate the signal in the same network before it becomes too weak. The purpose of regeneration is to extend the length over which the signal can be transmitted in the same network. There is no amplification involved but the regenerating of signal with original strength. It has two ports.
- **Hub:** It is a repeater with multiple ports. It is a connector like in star topology, which connects the wires coming from various branches. Hub cannot filter data and sends packets to all connected devices. Hubs are not designed to find the best path for data packet delivery.
Based on their features they are categorized into active, passive and intelligent hubs.
- **Bridge:** It is a repeater operating at data link layer. It can filter data by reading MAC addresses of sender and receiver. It can also be used to connect two LANs. It is a two port device.
It is of two type: transparent and source routing.
- **Switch:** It is a multiport bridge working on data link layer. It has capability of performing error checking before sending the packets.
- **Router:** It is a kind of switch that routes the data packets based on the IP address. It works on the network layer. It connects LANs and WANs. Router maintains the routing table based on which decision is taken for routing data packets.
- **Gateway:** It acts like a passage to connect two networks which may be working on different models. It act as data interpreter, that accepts data from one network, interprets it and send it to another network. It works on the network layer as protocol converter. It is more complex in design then switch or router.

Fig. 8 shows the media devices.

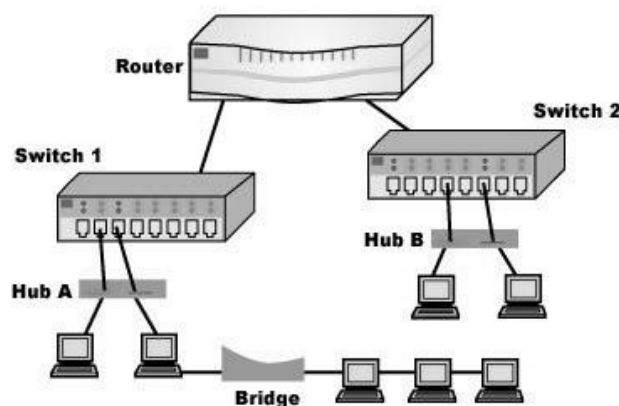


Fig 8: Transmission media devices

2.7 TRANSMISSION IMPAIRMENTS

The data communicated through transmission media deteriorates in the quality at the end of the media. The imperfection in transmission media causes transmission impairments.

There are three major causes of transmission impairments:

Attenuation: It is loss of signal strength. This is due to loss of signal energy in overcoming the resistance of transmission media. Amplifiers are used to amplify signals and compensate for the loss. Fig below shows the attenuation and amplified signal.

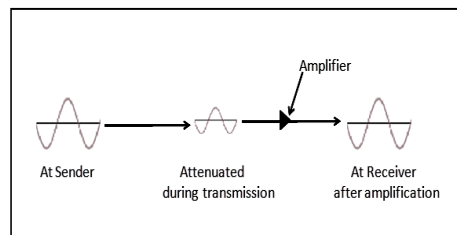


Fig 9: Attenuation in signal

Distortion: It is the change in the form or in the shape of signal. In a composite signal by combination of signals of multiple frequencies, each frequency component has different propagation speed. Due to this they reach at different timings at the receiver thus leading to distortion.

Fig below shows the change in phase of the received signal due to distortion.

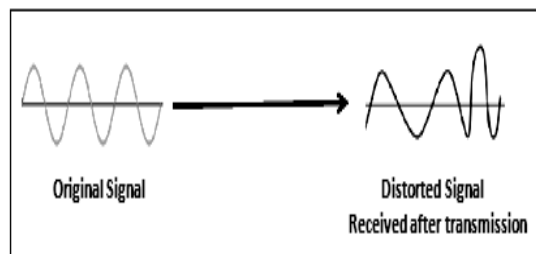


Fig 10: Distortion in signal

Noise: It is the unwanted signal that gets mixed with the original signal during transmission. It can be one of the following:

Induced noise: It comes from sources like appliances or motors.

Thermal noise: It is the movement of electrons in wires creating additional signal.

Crosstalk: It is due to one wire affecting another.

Impulse noise: It is high energy signal due to lightening or power lines.

These transmission impairments result in poor signal quality and resulting in low signal-to-noise ratio.

2.8 SUMMARY

The module covers the layered description of OSI model and TCP/IP model. OSI model is conceptual model whereas TCP/IP model is a practical model specifying how nodes can communicate in network and can be termed as compact version of OSI model. OSI model has

seven layers whereas TCP/IP model has four layers the functionality of some of the layers of OSI model has been combined into a single layer in TCP/IP model.

Physical layer covered in this module is the basis of all networks as it deals with the data transmission among two nodes. The functions major to the physical layer are multiplexing, switching, deciding transmission mode. The transmission media used for communication can be guided or unguided. It play an important role in defining channel bandwidth. The most commonly used guided media are twisted pair cables which are copper cables or optical fiber. Optical fiber are used for high speed communication over long distance as it has less attenuation, less prone to interference and durable as compared to copper wire.

For the physical connection of the node to network, network interface card is used. It can be active, passive or intelligent NIC card based on the functions performed.

Transmission media devices, repeater and hubs works on physical layer for regenerating signal. Hub is multiport repeater. Switches and bridges work on data link layer and routers work on network layer. Gateway acts as a translator between two networks deploying different protocols.

Transmission media used for communication has various imperfections resulting in signal attenuation, distortion and crosstalk.

2.9 REVIEW QUESTIONS

- a. In OSI model, how the data encapsulation occurs from top to bottom layer?
- b. In which layers the term frames and packets are used?
- c. Name some well-known protocols of application layer?
- d. Enumerate the differences between OSI model and TCP/IP model.
- e. List the major functions of all the seven layers of OSI model.
- f. Which layer in TCP/IP model performs the functions of session and presentation layers of OSI model?
- g. Illustrate the principle used in optical fiber for communication of data.
- h. Which features of optical fiber give it preference over copper wire for data transmission?
- i. How circuit switching is different from packet switching?
- j. What are the application areas of ethernet and wireless NIC?
- k. It is important in the network to uniquely identify the nodes for communication. How addressing helps in identifying the nodes and what kinds of addressing TCP/IP offers?
- l. Name the transmission media devices working on physical, data link and network layers respectively.

2.10 REFERENCES & FURTHER READING

- Data Communication & Networking by BehrouzForouzan
- Computer Networks by Andrew Tannenbaum
- Data and computer Communications by William Stallings
- Computer Networking: A Top-Down Approach by James Kurose and K.W. Ross

UNIT 3: DATA LINK LAYER

STRUCTURE

3.0 Objectives

3.1 Data Link Layer: An Introduction

3.2 Introduction

3.3 Services by Data Link Layer

3.4 Framing

3.5 Error Control

3.5.1 Error Detection and Correction

3.5.2 Error Correcting Codes

3.5.2.1 Hamming Codes

3.5.2.2 Binary Convolution Codes

3.5.2.3 Reed-Solomon Codes

3.5.2.4 Low Density Parity Check Codes (LDPC)

3.5.3 Error Detection Codes

3.5.3.1 Parity Check

3.5.3.2 Checksum

3.5.3.3 Cyclic Redundancy Check (CRC)

3.6 Flow Control Protocols

3.6.1 Simplex Protocols

3.6.1.1 Utopian Simplex Protocol

3.6.1.2 Simplex Stop-and-Wait Protocol for error free channel

3.6.1.3 Simplex Stop-and-Wait Protocol for a noisy channel

3.6.2 Sliding Window Protocols

3.6.2.1 One-bit Sliding Window Protocol

3.6.2.2 Sliding Window Protocol using go-Back-N

3.6.2.3 Sliding Window Protocol using Selective Repeat

3.7 Shared Media Protocols

3.7.1 Aloha

3.7.2 Carrier Sense Multiple Access (CSMA)

3.7.2.1 Non-Persistent CSMA

3.7.2.2 Persistent CSMA

3.7.2.3 CSMA with Collision Detection (CSMA/CD)

3.7.2.4 CSMA with Collision Avoidance (CSMA/CA)

3.8 Summary

3.9 Review Questions

3.10 References & Further Reading

3.0 OBJECTIVES

- Understanding the type of services provided by data link layer
- Understanding the framing techniques and the challenges in the schemes
- Getting introduced to error detection and error correction algorithms at data link layer
- Understanding the flow control mechanisms
- Understanding the media access protocols

3.1 INTRODUCTION

The data link layer is layer 2 in the OSI model. It uses the services of layer 1 i.e. physical layer for data transmission from sender to receiver. The functions of data link layer are:

- To provide service interface to network layer
- Dealing with the errors in the transmission
- Controlling the transmission flow to overcome the issues due to speed mismatch of sender and receiver

In this module we will focus on the concepts of frames, framing in data link layer, error control and flow control in data link layer. Also, protocols to share media for data transmission will be covered.

3.2 DATA LINK LAYER: AN INTRODUCTION

The data link layer is concerned with local delivery of frames between nodes on the same level of the network. Data-link frames, as these protocol data units are called, do not cross the boundaries of a local area network. Inter-network routing and global addressing are higher-layer functions, allowing data-link protocols to focus on local delivery, addressing, and media arbitration. In this way, the data link layer is analogous to a neighborhood traffic cop; it endeavors to arbitrate between parties contending for access to a medium, without concern for their ultimate destination. When devices attempt to use a medium simultaneously, frame collisions occur. Data-link protocols specify how devices detect and recover from such collisions, and may provide mechanisms to reduce or prevent them.

3.3 SERVICES BY DATA LINK LAYER

Data link layer offers the following services:

- **Unacknowledged connectionless service:** Frames are transmitted by data link layer to receiver without establishing any logical connection between sender and receiver. Frames received by receiver are not acknowledged. Ethernet is an example of this kind of service.
- **Acknowledged connectionless service:** In this service no logical connection is established but frames are acknowledged by the receiver. This service is mainly used in wireless systems where channels are unreliable.
- **Acknowledged connection-oriented service:** Sender and receiver establishes connection before transmitting the data. The frames sent are acknowledged by the receiver.

3.4 FRAMING

The data link layer uses the services of physical layer in transmitting data to the receiver in the form of bit stream of 1s and 0s. Due to noise in the wireless channel the error rate is high. Physical layer in order to bring the error rate within the permissible limits add redundant bits. It does not guarantee that the data received by data link layer will be error free. Data link layer detects and corrects the error.

To achieve this, layer breaks the bit stream into frames of fixed size, the checksum is performed for the frame. The checksum is also transmitted along with the frame data. At receiver side the checksum is computed. If checksum is same the frame is error free otherwise error is reported and data link layer takes measures to deal with the error.

It is important to understand how the breaking-up of bit stream is done to form the frames. It is important the start and the end of the frame is clearly known to the receiver. There are four methods for framing:

- **Byte Count:** It is the simplest way of framing. In the header information, count of bytes are specified which will make a frame. Receiver on receiving the frame checks the header information and count the number of bytes mentioned and consider that as one frame.

This is the simplest way of forming frames but it has a major drawback. In case the header information about count of bytes in a frame is transmitted erroneously, the receiver will not be able to identify the frame boundaries and will not be able to send message to the sender to retransmit as receiver cannot know that how many frames are to be asked for the retransmission.

The figure below specifies the frame sizes of 5, 5, 8 and 8 bytes in the data stream and shows the error in identifying the start and end of frame if byte count is garbled.

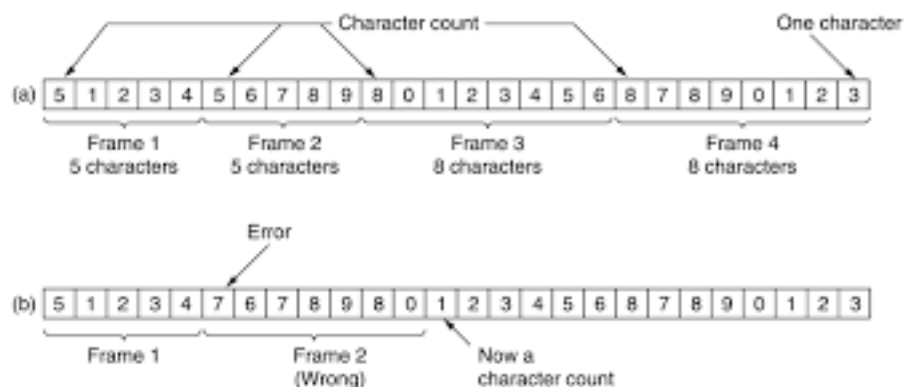


Fig1: Byte count framing a) byte stream without error b) byte stream with error

The figure clearly depicts that in case of error in the byte count value the start and end of frame as perceived by the receiver will be incorrect. This will result in erroneous data.

- **Flag byte with bit stuffing:** In this technique the start and end of the frame is identified by the adding special byte at the start and at the end of frames. This

special byte is known as flag byte. The flag byte is always same. Two consecutive flag bytes will tell that one frame has ended and another has started.

This will help receiver finding the start of new frame. The techniques faces a problem:

- If the flag byte occurs in the data itself then it would be wrongly interpreted. The solution to this problem is that data link layer of sender on identifying the same flag byte in the data should put special escape byte before the flag byte. Data link layer at the receiver on finding escape byte should remove it from the data. Adding escape byte is known as byte stuffing.
- Another problem arises when escape byte is also there in the data stream itself, in that case an additional escape byte is put by the sender. Receiver on getting the two escape bytes removes one from the data stream.

Fig 2 shows some of the examples of flag byte with byte stuffing.

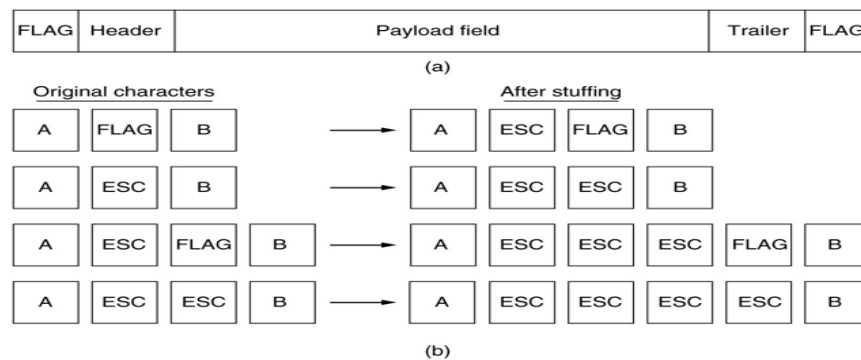


Fig 2: a) Frame format b) Byte stuffing with esc and flag bytes in data

- Flag bits with bit stuffing:** In flag byte, it becomes restriction to have one byte i.e. 8 bits for flag and for stuffing. This technique overcomes this drawback. This technique allows the frame to have multiple bits which may not be multiple of 8 to convert it into bytes. In this case each frame starts and ends with flag bits 01111110. If in the data five times digit 1 is identified by sender it automatically stuffs 0 in the bit stream. Receiver on identifying five 1s followed by 0, simply de-stuff the 0 and keep the remaining data. Fig 3 shows this technique. Fig. 3a shows the actual data at sender, 3b shows the bits stuffed by sender and fig. 3c shows the data at the receiver after removing stuffed bits.

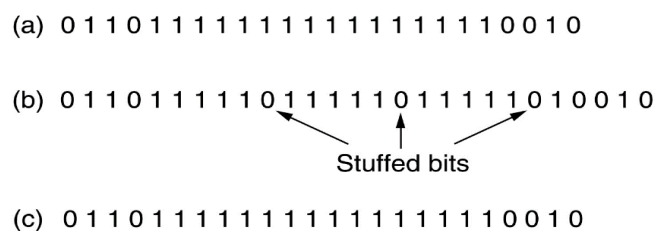


Fig 3: Flag bits with bit stuffing a) original message b) bit stuffing c) decoded message

Both the flag byte and flag bit techniques increase the size of the data stream due to byte and bit stuffing.

- **Physical layer coding violations:** This technique is applicable where during the physical transmission encoding is done in the network by adding redundant bits. In some cases 1 bit data is encoded into two bits, for eg. 1 is represented as high-low pair i.e. 10 and 0 is represented as low-high pair i.e. 01. This ensures that during transmission any bit there is transition from low to high or vice versa. Receiver can identify the bit boundaries in this case. The high-high and low-low combination is used for delimiting frames in some cases. In this technique no additional bit or byte stuffing is required.

Fig 4 shows this technique.

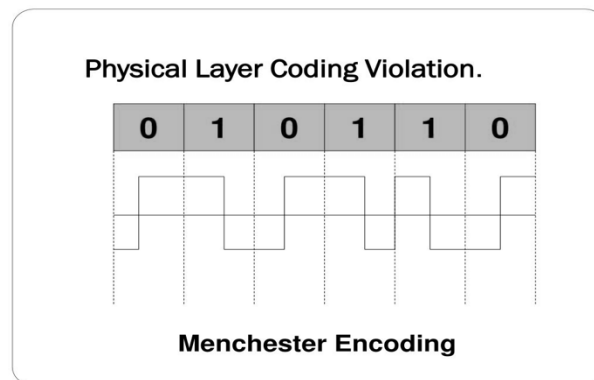


Fig 4: Technique for representing a bit as pair of two

All these techniques have their advantages and disadvantages. Many data link layer protocols use combination of these methods.

3.5 ERROR CONTROL

Once the framing is done the next challenge for the design link layer is to ensure that the frames are delivered to the network layer of the receiver in proper order and error free. Error corrections and error detection codes at data link layer ensure this.

Error control is not required for unacknowledged connection-less service but for acknowledged connection-oriented service it is required.

The positive or negative acknowledgement received by sender gives an indication for retransmission of frames in case of error. The problem occurs when the frame is lost and the acknowledgement is also lost. The sender will not receive the acknowledgement and hence keep on waiting infinitely. This can be handled by starting a timer once the frame is transmitted, if no acknowledgement is received before timer expires, sender re-transmits it.

3.5.1 Error detection and correction

Error correction schemes are used in data link layer to rectify the errors in the transmission due to noise or interference. Two basis strategies have been devised for error detection and error correction:

- To include enough redundant bits in the transmission so as to enable the data link layer at receiver to identify error and deduce the actual transmitted data from it. This strategy uses error-correction codes for detecting and correcting errors.
- To include redundant bits only to indicate receiver that error has occurred so as to send request for re-transmission. This strategy uses error detection codes.

The use of error correcting codes is referred to as forward error correction (FEC). The error correction codes are used in case the transmission media is unreliable but for fiber optics where the chances of errors in transmission are very low, error detection and retransmission of frames is comparatively cheap.

To understand the error handling technique it is important to understand the how error is identified. A frame consists of m bits of data and r redundant bits. Total message bits will be $m+r$, which can be denoted by n . This n -bit data unit with message bits and redundant bits for checking is known as codeword.

By applying logical operation exclusive OR (XOR) it can be identified that how many bits differ in the actual codeword and received codeword. Eg.

Original codeword	10100101
Received codeword	10111001
XOR Operation	00011100

XOR gives the count of bits that have been erroneously received. In this example 3 bits are incorrect. Number of bit positions in which the two given codewords vary is termed as hamming distance. Hamming distance in this example is 3. The hamming distance tells that how many single bit errors are required to exactly match the one codeword to another. The error-detecting and error correcting properties of a code depend on its Hamming distance. To detect the errors in d data bits a code with distance $d+1$ is required as with such code there is no chance that d single bit errors can change valid codeword to any other valid codeword. To correct errors in d data bits, a code with distance $2d+1$ is required.

3.5.2 Error Correcting Codes

There are four error correcting codes that are commonly deployed for data link layer.

3.5.2.1 Hamming Codes: It is a block code capable of correcting single bit errors and errors upto two simultaneous bits. In this method, encoding by sender is done by adding redundant bits in the message. These are the extra bits added at specific positions in the message. At receiver, the logic is applied to detect errors and find the erroneous bit positions. The process of encoding the message using Hamming code involves the following basic steps:

- **Calculating the count of redundant bits:** For m bit message, r redundant bits are added. So total bits in encoded message will be $m+r$. Considering error can be in any of the $m+r$ bits, there can be $m+r$ different states of code words and one state with no error. In total different states of code word will be $m+r+1$. In a binary number system r bits can have 2^r states so the following equation holds $2^r \geq m+r+1$.
- **Positioning of redundant bits:** The r redundant bits are placed at positions $r_1, r_2, r_4, \dots, r_n$, where these positions are defined as power of 2, i.e. 1, 2, 4, 8... so, $r_1 = 1, r_2 = 2$, and so on.
- **Calculating the values of redundant bits:** These redundant bits are parity bits, generally even parity is taken. Even parity is when the count of 1s in binary number is even. The redundant bits are calculated by checking the even parity.
 r_1 is the parity bit for all the data bits whose binary representation includes 1 in the least significant position i.e. 1,3,5,7,9,11...etc.
 r_2 is the parity bit for all the data bits whose binary representation includes 1 in the second position from the least significant bit i.e. 2, 3, 6, 7, 10, 11...etc
 r_4 is the parity bit for all the data bits whose binary representation includes 1 in the third position from the least significant bit i.e. 4-7, 12-15, 20-23...etc

r_8 is the parity bit for all the data bits whose binary representation includes 1 in the fourth position from the least significant bit i.e. 8-15, 24-31, 40-47...etc

The encoded message by adding redundant bits is transmitted by sender. At receiver the message is decoded. The following are the decoding steps:

- Count and position of redundant bits is ascertained using the same logic as in encoding.
- **Parity Checking:** Parity bits are checked based on the data and redundant bits using the same rule as for generating parity bits.
 c_1 is parity for bits at position 1,3,5,7,9,11....
 c_2 is parity for bits at position 2,3,6,7,10,11....
 c_3 is parity for bits at position 4-7,12-15, 20-23....
 c_4 is parity for bits at position 8-15, 24-31,40-47....

Decimal equivalent of $c_1 c_2 c_3 c_4$ is calculated. This value represents the location of erroneous bit in the message. To correct the same, the bit is flipped.

Following example explains this method:

Eg.

Data bits to be transmitted are 1011010. Using the equation $2^r \geq m+r+1$, r is calculated as 4. So four redundant or parity bits are added at specific locations

11	10	9	8	7	6	5	4	3	2	1
1	0	1	r_8	1	0	1	r_4	0	r_2	r_1

r_1 =parity of 1,3,5,7,9,11 bits, count of 1s at these locations is odd excluding r_1 , so make it an even parity value of r_1 has to be 0.

r_2 =parity of 2,3,6,7,10,11 bits, count of 1s at these locations is even excluding r_2 , so to make it an even parity value of r_1 has to be 0.

r_4 =parity of 4-7bits, count of 1s at these locations is even excluding r_4 , so to make it an even parity value of r_4 has to be 0.

r_8 =parity of 8-11bits, count of 1s at these locations is even excluding r_8 , so to make it an even parity value of r_8 has to be 0.

The encoded data will become

11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	1	0	1	0	0	0	0

Let the transmitted data be, where 9th bit is erroneous bit

11	10	9	8	7	6	5	4	3	2	1
1	0	<u>0</u>	0	1	0	1	0	0	0	0

Receiver will decode the data and calculate

c_1 is parity for bits at position 1,3,5,7,9,11 , which is odd so $c_1=1$

c_2 is parity for bits at position 2,3,6,7,10,11, which is even so $c_2=0$

c_3 is parity for bits at position 4-7, which is even so $c_3=0$

c_4 is parity for bits at position 8-11, which is odd so $c_4=1$

$c_1 c_2 c_3 c_4$ is 1001, decimal equivalent of it is 9. Hence error at 9th bit is detected, value of 9th bit is changed from 0 to 1.

Hamming code can correct one bit error but can detect two bits error.

3.5.2.2 Binary Convolution Codes :It is different from the block code. Encoder processes a input bits sequence and generates output bits sequence. These codes are commonly used for mobile phone system, satellite communications etc.

The encoder is collection of shift registers through which the input bits are sequentially passed.

Convolution code is represented by (n, k, K) , i.e. for every k bits, there are n output bits and K being the constraint length. K is the count of shift registers in the encoder.

Shift registers in this encoder has memory, so the output of n bits not only depend on the k input bits but also on previous $K-1$ blocks each of k input bits, giving output bits n as $K \times k$. Generally n and k are small numbers. It is easy to implement and is effective in communication where the error rate probability is high and channels are noisy.

3.5.2.3 Reed-Solomon Codes: It is also a kind of linear block codes like hamming codes. Instead of operating on individual bits, it works on m bit symbols. These codes work on the fact that n+1 points are required to define any polynomial of degree n. For eg. polynomial ax^2+bx+c , of degree two can be defined by three points a, b and c. Extra points are redundant and can be utilized for error correction. To correct the error in addition to the data points additional points satisfying the polynomial equation are transmitted. If there is error in any point, still the polynomial equation can be recovered by using other points. For m bit symbol, the codeword will be $2^m - 1$ symbols long. If m=8 bytes, then the codeword will be of 255 bytes. The most widely used code is (255, 233) code, i.e. 32 redundant symbols are added to 233 symbols of data. It can correct errors upto 16 symbols or 128 bits. Decoding with error correction is done with an algorithm. In general on adding 2n redundant bytes it can correct n bytes of errors.

It is used for correcting burst errors, which is error in the contiguous sequence of symbols. These codes are used for error correction in satellite communications, data over cables, CDs, blu-ray discs etc.

3.5.2.4 Low Density Parity Check Codes (LDPC): These are linear block codes. Every single output bit is formed from a fraction of input bits. These codes are specified by parity matrix, containing mostly 0s and low density of 1s, hence the name LDPC. It is useful for error corrections in large blocks of data. It is mainly used in power line networks, digital video broadcasting etc.

3.5.3 Error Detection Codes

For wireless transmission media where the channel reliability is low and transmission is prone to errors, the error correction is preferred. In case of reliable channel, eg. using optical fiber or copper cable as transmission media, error detection and retransmission is efficient mechanism. There are three linear systematic error detecting block codes:

3.5.3.1 Parity Check: This check is done by adding extra bit, termed as parity bit to make count of 1s even in data in case of even parity. Sender count 1s in the data and set parity bit as 1 if no of 1s counted are odd. Receiver on receiving the data counts 1s in the received data, if count is even it is accepted else rejected. It can only detect single bit error.

Eg. data to be transmitted is 1001010, the parity of the data is odd so to make it even parity bit is added with value 1, the data 1001010 1, including parity bit is transmitted. At receiver end parity is again checked and in case there is change in parity from even to odd, the error is detected and data is discarded.

Parity check can be done in matrix of data, where parity is calculated for all columns and rows and parity bits are also transmitted along with data.

Eg.

The following data is to be transmitted

1000101	1110001	1010110	0111001

Parity is checked

Row Parity

1000101	1
1110001	0
1010110	0
0111001	0
1011011	1

Column Parity

The following data is transmitted by sender, where the underlined digits are the parity bits. Any change in parity at the receiver side is detected as an error.

1000101 <u>1</u>	1110001 <u>0</u>	1010110 <u>0</u>	0111001 <u>0</u>	<u>1011011</u>

3.5.3.2 Checksum

This scheme uses group of parity bits to detect errors. In this case data is divided into fixed size blocks. At sender side, segments are added using 1's complement arithmetic to get the sum. Complement of sum gives the checksum.

The checksum segment is transmitted along with the data. At the receiver all the segments are added using 1's complement and further the sum is complemented. All zeros tells no error hence the data is accepted, otherwise the data is rejected and request for retransmission is sent.

Eg. the following data is to be transmitted

10110011	10101011	01011010	11010101

If $k = 4$, and $n = 8$ then

$$\begin{array}{r}
 k=4, \quad n=8 \\
 10110011 \\
 10101011 \\
 01011010 \\
 \hline
 01011110 \\
 \text{Carry} \rightarrow 1 \\
 01011111 \\
 01011010 \\
 \hline
 10111001 \\
 11010101 \\
 \hline
 10001110 \\
 \text{Carry} \rightarrow 1 \\
 \text{Sum : } 10001111 \\
 \text{Checksum } 01110000
 \end{array}$$

At sender side

$$\begin{array}{r}
 10110011 \\
 10101011 \\
 \hline
 01011110 \\
 \text{Carry} \rightarrow 1 \\
 01011111 \\
 01011010 \\
 \hline
 10111001 \\
 11010101 \\
 \hline
 10001110 \\
 \text{Carry} \rightarrow 1 \\
 10001111 \\
 01110000 \\
 \hline
 \text{Sum: } 11111111 \\
 \text{Complement} = 00000000 \\
 \text{Conclusion} = \text{Accept data}
 \end{array}$$

At receiver side

Fig 5: Calculation of checksum at sender and at receiver side

3.6 FLOW CONTROL PROTOCOLS

Flow control is technique to synchronize the speeds of sender and receiver. In data link layer, flow control restricts the number of frames the sender can transmit before it gets the acknowledgement from the receiver. Flow control protocols in data link layer uses two techniques:

- **Stop-and-wait:** In this technique, sender transmits the frame and wait for acknowledgement. Once it gets the acknowledgement from receiver, sender sends the next frame.
- **Sliding Window:** In this technique, both the sender and receiver have buffer of finite size called window. Buffer size is fixed on mutual consent of sender and receiver. Sender transmits multiple frames in a sequence without waiting for acknowledgement till the time window is filled. Then it waits for acknowledgement. On getting acknowledgements it slides the window based on the number of acknowledgements received and start transmitting the next frames in the sequence.

3.6.1 Simplex Protocols

3.6.1.1 Utopian Simplex Protocol

This protocol assumes that there is infinite buffer size on both sender and receiver side. This is an ideal protocol and not realistic that's why the name Utopian has been given. It assumes a single direction transmission with no transmission error.

Sender transmits the data at a fast speed and the receiver receives it. This protocol assumes that there is no requirement of error control and flow control.

3.6.1.2 Simplex Stop-and-Wait Protocol for error free channel

Simplex protocol suffers from drawback of flooding receiver with frames at a rate faster than it can process. It takes into account the limitation on processing speed of receiver. Due to this limitation the frames will be dropped at receiver end. In this protocol as error free channel is considered so no retransmissions are considered. The sender transmits the frame, waits for acknowledgement, once received it transmits next frame.

3.6.1.3 Simplex Stop-and-Wait Protocol for a Noisy Channel

This protocol considers the realistic scenario where error control and flow control are required. It assumes that the processing speed of receiver is limited and frames may get corrupted while transmission. If the sender speed is high and receiver is not able to process, the frames get dropped. Also, the frames with error are discarded by data link layer of the receiver. Receiver sends acknowledgement only for the valid frames. After getting acknowledgement sender transmits other frames and also resends the frames for which no acknowledgment has been received.

Sender add the sequence number to frame and transmits it. After receiving positive acknowledgement it transmits next frame. In case of no acknowledgement it waits for fixed amount of time and resends the frame.

Receiver maintains the sequence numbers of frames. It matches the sequence number of the frame received with the expected sequence number and sends acknowledgement which also has a sequence number.

This protocol is also known as simplex stop-and-wait Automatic Repeat Request protocol for noisy channel. Fig. 7 represents the various scenario of this protocol.

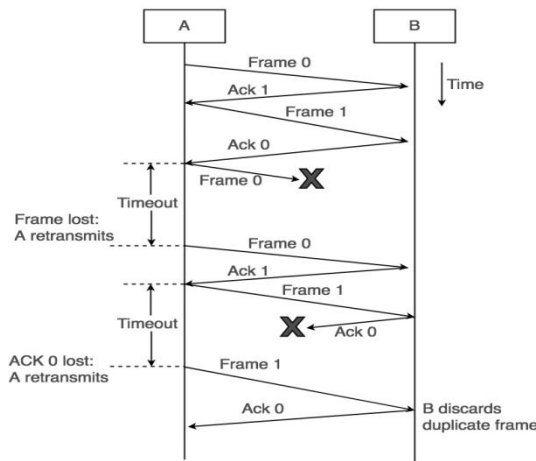


Fig 7: Simplex stop-and-wait Protocol for nosy channel

3.6.2 Sliding window Protocols

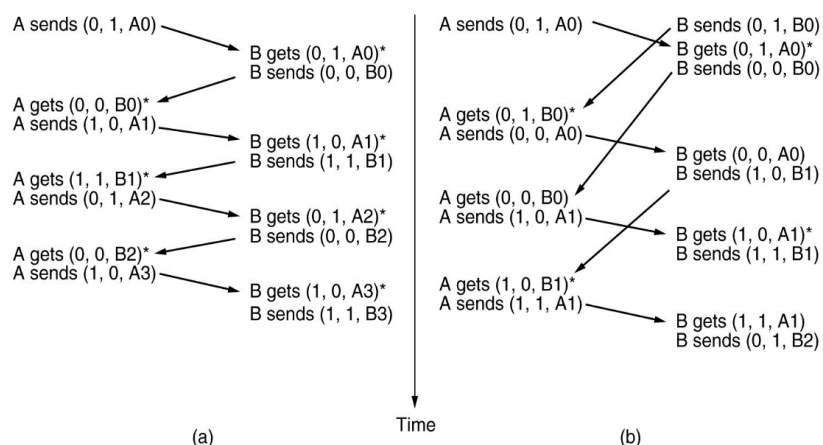
Simplex protocols have one way transmission mode. The bidirectional transmission which is full duplex transmission adds complexity in the protocols. Piggy-backing technique helps in achieving full duplex transmission whereas acknowledgement is sent along with the frame to be transmitted. Sliding window protocols are bi-directional protocols. Following are the types of sliding window protocols for data link layer:

3.6.2.1 One-bit sliding window Protocol

Referring to the sliding window technique, in this protocol the window size is one bit. In this protocol sender transmits frame, wait for acknowledgement, once received it moves its window to next and transmits next frame. It is similar to stop-and wait protocol.

For full duplex transmission, the acknowledgement is piggy-backed with the frame. The data frame to be transmitted has additional field for acknowledgement (ack). The ack field contains sequence number of last frame received without error. If this sequence number matches with the sequence number of next frame to be transmitted, then no error is reported otherwise it is assumed that there was error in earlier transmission.

Fig 8 a) represents ideal case of the protocol where notation (seq no., ack, frame number*) has been used. The asterisk specifies that the frame has been received. Fig. 8 b) represents the simultaneous transmissions resulting in half of the frames transmitted are duplicates. It can also result due to early timeout, causing same frame to be transmitted multiple times.



This protocol assumes sender window of size **N** and receiver window of size **1**. It is also known as go-back-n ARQ protocol. The frames in this case are numbered sequentially. Sender transmits frames depending on the window size without waiting for acknowledgement. If sender does not get the acknowledgement of frame within specified time, so all the frames starting from that frame are re-transmitted. If the sending window size is represented as 2^n-1 , then n bit sequence number can be applied for numbering frames where sequence number will be assigned from 0 to 2^n-1 .

The diagram illustrates the Stop-and-Wait protocol. It shows a sequence of frames (0-9) and acknowledgments (Ack 0-7). A 'Timeout interval' is marked between frame 4 and frame 2. An 'Error' occurs with frame 3, leading to 'Frames discarded by data link layer' (frames 4-8).

Go-back-n protocol has drawback of retransmitting all the frames after the erroneous frame. In case errors are rare and channel is reliable, this protocol has acceptable performance but in case the errors are more it will result in retransmitting multiple frames, which will take lot of bandwidth.

In selective repeat protocol sender and receiver has specific window size and receiver can accept frames out of order. The frames are buffered at receiver and is shared by data link layer to upper layer.

In case of damaged frame a non-acknowledgement of the frame only the selected frame is to be retransmitted.

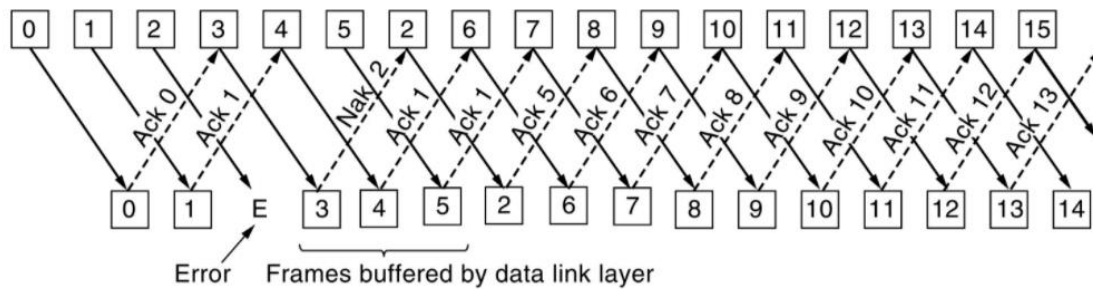


Fig 10: Sliding Window elective Repeat with large window size

Fig. 10 shows the working of selective repeat protocol. The window size both at receiver and sender side is large. The non-acknowledgement of frame 2 tells sender to retransmit only that frame.

Sliding window protocols have advantage over stop-and -wait as these are more efficient. Moreover, selective repeat protocol avoid unnecessary retransmissions of frames.

3.7 SHARED MEDIA PROTOCOLS

The sublayer of data link layer, termed as Medium Access Control (MAC) layer. One of the major functions of MAC layer is to perform multiple access resolutions and to determine the channel access method for data transmission. In case of multipoint communication, multiple devices can access the transmission media or channel simultaneously. To avoid the collision and crosstalk, multiple access protocols are required to define rules for the sharing of communication media.

A category of shard media protocols is random access protocols. There is an assumption that all the devices have equal priority to send the data over communication media. The main protocols under this category are as follows.

3.7.1 Aloha

There are two categories of aloha protocol, pure aloha and slotted aloha.

Pure aloha protocol works on the following rules:

- Any node/device can transmit data through channel at any time without considering the status data transmission by other devices.
- Data frames lost in collision are to be retransmitted based on the acknowledgement mechanism. If no acknowledgement is received within the specified time, the node waits for random amount of time termed as backoff time and retransmits the frame.

The efficiency of this protocol is very low.

Slotted aloha protocol was designed to enhance the efficiency of pure aloha. In this protocol, shared channel is divided into slots of fixed time duration. Node can only transmit at the beginning of time slot and only one frame can be transmitted per slot. This helps in avoiding collision to some extent.

3.7.2 Carrier Sense Multiple Access (CSMA)

It is possible in LAN to sense the carrier i.e. transmission media and act accordingly. Protocols using the mechanism of carrier sensing are CSMA category protocols. If node wants to transmit data, it senses the media and if channel is found idle, it transmits. There are various access modes in CSMA.

3.7.2.1 Non-persistent CSMA

The node senses the media, if not idle then it waits for a random time and rechecks the media. This process continues till the node finds the media idle and then it transmits.

3.7.2.2 Persistent CSMA

1-persistent: The node senses the media, if not idle then it keeps on checking continuously till it finds the media idle and then transmits with full probability of 1.

P-persistent: This is different from 1-persistent, in a way, that when the media is found idle, node transmits the frame with p probability and not with 1 probability. With probability $1-p$ it defers the transmission for the next time slot.

3.7.2.3 CSMA with Collision Detection (CSMA/CD)

Persistent and non-persistent CSMA protocols have better performance as compared to aloha but if the two nodes begin transmission simultaneously then there will be collision.

CSMA/CD protocol implements a mechanism of collision detection. If any collision is detected, the node sends stop signal to the shared media channel to terminate transmission. After that, the node waits for a random period of time before next transmission. CSMA/CD consists of transmission, contention and idle period. After transmission node will contend for the shared media, once it finds the media idle it will transmit. If there is no data to transmit it will be in idle state. Fig. 11 shows the three status of nodes in CSMA/CD.

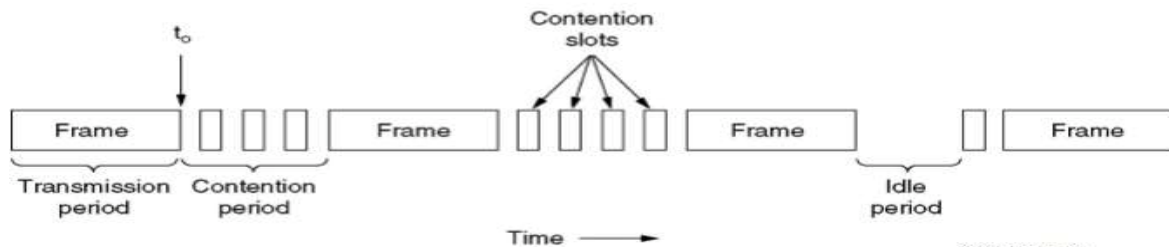


Fig 11: Status of nodes in CSMA/CD

3.7.2.4 CSMA with Collision Avoidance (CSMA/CA)

The protocol focuses on collision avoidance by using the signalling mechanism. In this protocol, before transmitting the actual frame, sender node transmits request to send (RTS) frame indicating the duration of transmission using network allocation vector (NAV), which is broadcasted to all the channels.

Receiver node replies with clear to send (CTS) frame. It indicates the hidden nodes as well. Once CTS is received, sender node will start transmitting.

NAV is used for virtual carrier sensing and used in IEEE 802.11 standard of wireless LAN. The duration of transmission is specified so the other nodes know for how much time they need to defer accessing the shared media. Zero value of NAV indicates the media is idle.

Fig. 12 shows the working of CSMA/CA protocol

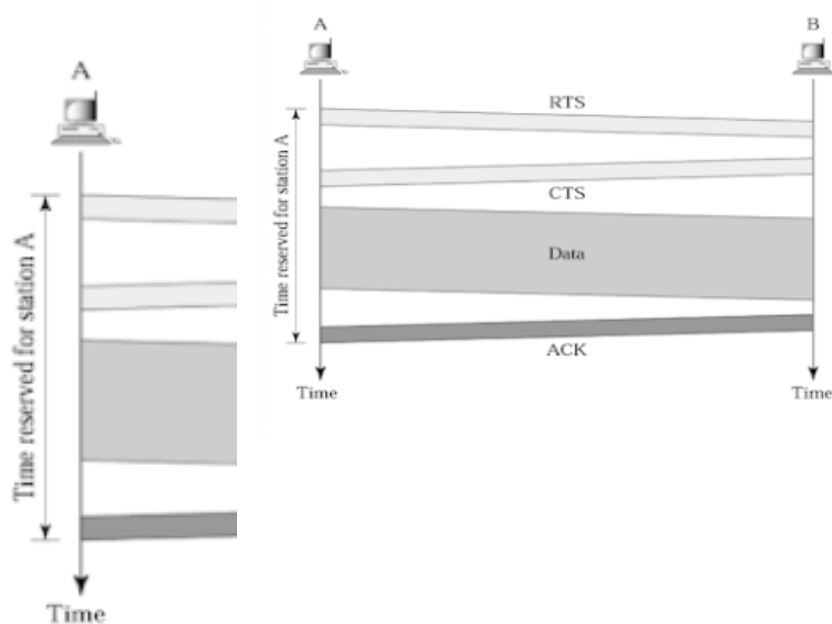


Fig 12: Working of CSMA/CA

3.8 SUMMARY

This module focuses on the functionality of data link layer. The data unit is frame and various techniques can be applied to help identifying the frame boundary at receiver. The error detection and error correction mechanism in data link layer help in attaining reliable communication. Wireless media is more prone to errors so error correction is not an efficient option, instead, error is detected and then retransmission request is send.

Flow control is important to avoid receiver being choked with data which then is discarded by receiver. Stop-and-wait protocols following half duplex communication mode has low efficiency as compared to sliding window protocols.

MAC sublayer, which is the lower among the twosublayers of data link layer deals with the access of the share media for communication. MAC protocols help node attaining control over the shared media to transmit frame with less probability of collision. CSMA/ CA is preferred over CSMA/CD though the overheads are more.

3.9 REVIEW QUESTIONS

- 1.12 What are the major functions of data link layer?
- 1.13 Specify the reasons for using error correction codes instead of error detection and retransmission.
- 1.14 Mention the two broad categories of flow control in data link layer and discuss about their efficiencies.
- 1.15 Which techniques of framing can result in the most number of stuffing bits in the frame?
- 1.16 Define Hamming distance. How many bits of errors can hamming code correct?
- 1.17 With an example, explain how checksum and CRC works.
- 1.18 If the generator polynomial is $1x^4+0x^3+0x^2+ 1x^1+1x^0$, identify if the message transmitted 1101011111110 has error or not.
- 1.19 Compare the performance of sliding window go-back-n and selective repeat protocols.
- 1.20 In parity check technique, find the row and the column parity of data, when both sender and receiver agree for odd parity.

1000111	1110101	1000110	0101001

- 1.21 Identify the suitable media access protocol for wireless transmission media and specify the reason for the choice.

3.10 REFERENCES & FURTHER READING

- Data Communication & Networking by BehrouzForouzan
- Computer Networks by Andrew Tannenbaum
- Data and computer Communications by William Stallings
- Computer Networking: A Top-Down Approach by James Kurose and K.W. Ross

$x^5 + x^4 + x^2 + 1$

In CRC assume the generator polynomial $G(X)=x^5 + x^4 + x^2 + 1$ is to be used. Determine whether the received message 111000110101110 has detectable errors or not. Show all workings.

$x^5 + x^4 + x^2 + 1$

In CRC assume the generator polynomial $G(X)=x^5 + x^4 + x^2 + 1$ is to be used. Determine whether the received message 111000110101110 has detectable errors or not. Show all workings.

UNIT 4: NETWORK LAYER

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Network Layer: An Introduction

4.3 Services Provided by Network Layer

4.3.1 Connection Oriented Service – Virtual Circuits

4.3.2 Connectionless Service – Datagram Service

4.3.3 Comparison of Virtual Circuit and Datagram Approach

4.4 IP Address

4.4.1 Understanding IP Addressing

4.4.2 Classes of IP Addresses

4.4.3 Network Masking

4.5 Subnetting

4.6 Routing Algorithms

4.6.1 Type of Routing Algorithms

4.6.2 Non-adaptive Algorithms

4.6.2.1 Flooding

4.6.2.1.1 Controlled Flooding

4.6.2.1.2 Selective Flooding

4.6.2.2 Random Walk

4.6.3 Adaptive Algorithms

4.6.3.1 Centralized Algorithm

4.6.3.2 Isolated Algorithm

4.6.3.3 Distributed Algorithm

4.6.3.4 Distance Vector and Link State Packet Routing Algorithms

4.6.3.5 Comparison of Distance Vector Routing and Link State Routing

4.7 Summary

4.8 Review Questions

4.0 OBJECTIVES

- a. Understanding the virtual circuit and datagram services provided by network layer
- b. Understanding IP addressing technique
- c. Identifying the subnet masks is IP addressing
- d. Understanding the protocol functioning in network layer
- e. Analysing the difference between various protocols

4.1 INTRODUCTION

Network layer is responsible for end-to-end delivery of packets from sender to receiver. The transmission may be single hop or multi-hop where intermediary nodes termed as routers are involved for the transmission. The major function of the network layer is to decide the transmission route. For this network layer should be aware of the network topology for choosing the right path. Network layer deals with the problems in defining routes for communication when both sender and receiver are on same network or on different networks. This module focuses on two kind of services provided by network layer connection-oriented i.e. virtual circuit and connectionless i.e. datagram. Also, IP addressing and routing protocols will be discussed in the module.

4.2 NETWORK LAYER : AN INTRODUCTION

The network layer is a portion of online communications that allows for the connection and transfer of data packets between different devices or networks.

The network layer is the third level (Layer 3) of the Open Systems Interconnection Model (OSI Model) and the layer that provides data routing paths for network communication. Data is transferred to the receiving device in the form of packets via logical network paths in an ordered format controlled by the network layer.

4.3 SERVICES PROVIDED BY NETWORK LAYER

Network layer provides the services to transport layer with the goal that transport layer should be independent of router technology and network topology. Also, the network addresses provided by network layer should be unique and uniform. Data link layer deal with frames and network layer deals with packets as a data unit for transmission.

Network layer is the lowest layer in OSI model to provide end-to-end transmission. It provides the following two types of services:

4.3.1 Connection-Oriented Service – Virtual Circuit

In this type of communication service a path from the source router to the destination router is established before sending the data packets. The connection establishes is known as Virtual Circuit (VC). Eg.the telephone system is a VC network.

The purpose of virtual circuit is to avoid the process of choosing new path every time the packet is to be transmitted. Every intermediary node i.e. router store the path established between source router and destination router in its memory in the form of routing tables. Same route is followed for transmitting all the packets from the defined source to defined destination. Once the transmission is complete VC is terminated.

In this service, each packet carrier a identifier specifying the virtual circuit to which it belongs.

Fig 1, details the concept of virtual circuit and routing tables.

Sender host H1 wants to transmit data to receiver host H2. Connection from H1 to H2 is established and given an identifier 1.

Data packet is transmitted from H1 to connected router i.e. A. A maintains a routing table which checks that if packet from H1 comes with a identifier 1, then transmit to router C. C in its table checks that any packet coming from A with identifier 1 is to be routed to E. E in its table checks that any packet coming from C with identifier 1 has to be transmitted to F.

Router F is connected to receiver host H2, so transmits data. Identifier 1 tells H2 that data has been transmitted from sender host H1.

In case another sender host H3 wants to transmit data to H2, it sends a packet to router A keeping the identifier as 1. Router A on checking the packet adds an entry in routing table to transmit this packet to router C but with identifier 2 as identifier 1 has already been assigned to packet received from other host.

C checks in the table that any packet received from A with identifier 2 is to be routed to router E with identifier 2 and further E in its table checks and routes the packet to F with identifier 2.

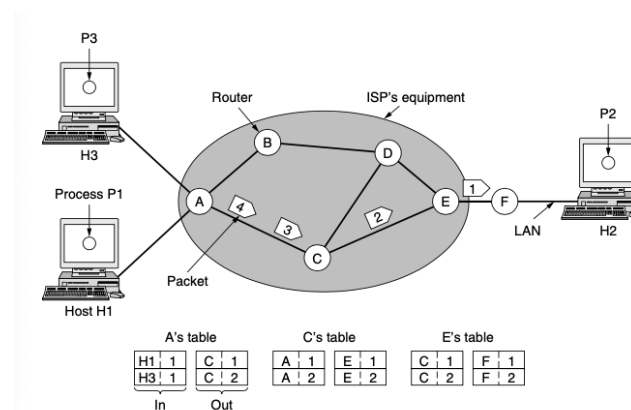


Fig 1: Defining path in Virtual Circuit

Router F transmits it to receiver host H2 and the identifier specifies that packet has been transmitted from sender host H3.

Router decides the path further based on the congestion and traffic in the network.

4.3.2 Connectionless Service – Datagram Service

In a connectionless service packets from sender host is numbered and transmitted through the available route. There is no advance connection setup like in virtual circuit. In this type of service packets are called **datagrams** and the network is termed as **datagram network**.

Fig 2 shows that the process in sender host H1 wants to communicate to process p2 at receiver host H2. The complete message shared by transport layer of H1 is divided into 4 packets of equal size.

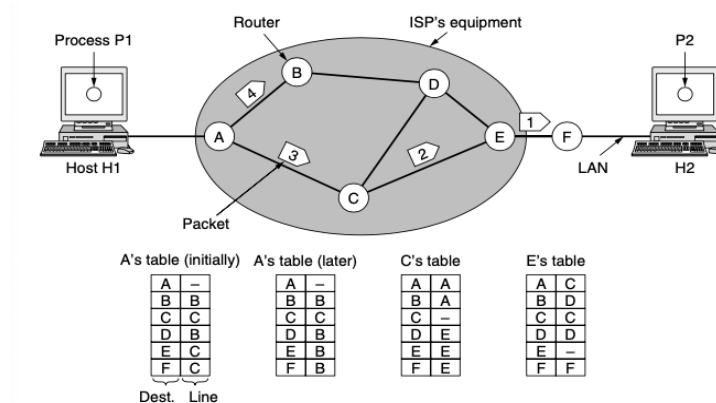


Fig 2: Datagram Service

H1 transmits each of the four packets to router A. A has internal routing table where the next router in the path i.e. line is specified based on the destination. In routing table of A there are only two outgoing paths i.e. through B and through C which are directly connected to A through point-to-point protocol.

Every packet received by A can be further transmitted to either B or C. After verifying the checksum for errors, A sends packet 1,2 and 3 to C, which further sends them to E and E sends packets to F which is further delivered to H2.

After transmitting three packets from same route, router A probably learned from the network that the path is congested, so routed it through another path. This decision is taken by routing algorithms.

4.3.3 Comparison of Virtual Circuit and Datagram Approach

Both virtual circuit and datagram approach has its benefits and drawbacks. Their comparison on the key points is detailed as below:

Key Point	Virtual Circuit Approach	Datagram Approach
Setting up circuit	Required	Not Required
Routing	Routing path is fixed during the set-up of VC and packets follow this path	Packets are routed independently, no fixed path is there
Packet delivery	All packets reach successfully as dedicated circuit is provided	Reliability of packet delivery is less as compared to VC
Effect of router failure	All VCs that pass through the failed router are terminated	No effect as packets can be routed from different path
Cost and Complexity	VC involves additional cost	Datagram network is

	of circuit set-up so it is more expensive but it is less complex	comparatively cheaper but is more complex
Example	Asynchronous Transfer Mode (ATM) used for telephone network	IP network for Internet services

4.4 IP ADDRESSES

An IP address is a unique address assigned to the device on internet. There are two versions of IP address; IPv4 and IPv6. IPv4 is a 32 bit address that can uniquely identify 2^{32} devices on the internet, whereas IPv6 is a 128 bit address and can uniquely identify 2^{128} devices.

4.4.1 Understanding IP addressing

Considering IPv4, 32 bits are divided into two parts i.e. host address and network address with the help of subnet mask. 32 bits address is written as four octet address, where 1 octet is 8 bits. Each octet is converted into decimal number and is separately by dot from other octet. The decimal value of octet is from 0 to 255 and in binary it will be from 00000000 to 11111111, where both values are inclusive. Eg. 192.168.52.100 is a representation of an IP address.

4.4.2 Classes of IP Address

IP addresses are divided into five classes from class A to class E but now a days the thisclassful addressing is rarely used. Fig. 3 specify the how this addressing works. In class A, first octet is for network address and remaining three octets i.e 24 bits for host addresses. These bits can be divided into subnets by the network administrator. Class A can have 2^7 i.e. 128 networks and 2^{24} i.e. 16 million hosts. The address range as given in figure is 1.0.0.0 to 127.255.255.255.

Class B can have 2^{14} i.e 16384 networks and 2^{16} i.e. 65536 hosts. Class C has 2^{21} networks and 2^8 i.e. 256 hosts. Class D is for multicast and class E is for future use. The address range for each class has been specified in the figure.

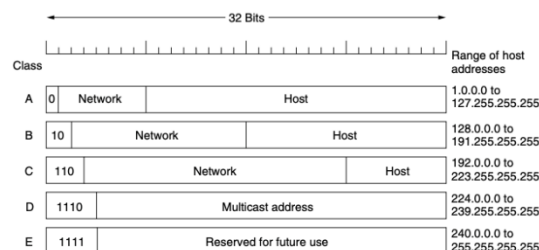


Fig. 3: Classful Addressing of IP Addresses

Considering the need for allocation of only required or limited number of Ips classless interdomain routing (CIDR) concept was introduced. Whenever user asks for specific number of IP addresses, CIDR dynamically assigns the block of IP addresses which are contiguous addresses and are in power of 2. The notation used for CIDR is for eg.

192.168/52.100 / 28. Number 28 signifies that 28 bits are being used for identifying network and remaining 4 bits are for identifying the host.

2.4.3 Network Masking

Network mask helps in identifying as to which part of address bits identifies network and which part identifies host. Class A, B and C has default masks as their network bits are fixed.

Following are the masks for the classes:

Class A : 255. 0. 0. 0

Class B : 255. 255. 0. 0

Class C : 255. 255. 255. 0

Masking helps in identifying network address. For eg.

Address 21.10.30.16 is a class A address with a subnet mask 255.0.0.0

In binary form it can be written as

Address : 00010101. 00001010 .00011110 . 00010000

Mask : 11111111. 00000000 .00000000 . 00000000

Octet of IP address corresponding to Mask with all 1s is network address and other is host address. We get network id in binary as 00010101 and host id as 00001010. 00011110. 00010000 in decimal network address is 21.0.0.0 and host address is 0.10.30.16.

4.5 SUBNETTING

Subnetting allows to create multiple logical networks within the specified class A,B or class C networks. In any organization the large network adds complexity and result in traffic congestion. Creating a subnet allows to put limit on number of routers for communication hence results in faster communication. The subnetting provides the following benefits:

- Broadcast domain can be divided resulting in efficient data communication and improved network performance.
- Data Packets need to travel a shorter distance to reach to the destination leading to efficient communication.
- Network security can be improved, the sensitive data can be transmitted through a specific subnet only. It gives more control to the administrators to have control over data transmission.

If the major networks in class A, B or C is divided into smaller subnetworks, it allows to create smaller interconnecting networks. Each device will have unique network/subnetwork id.

Subnetting can be achieved by extending the standard masking with some bits of host id part. Eg. Consider class C network with address 205.17.13.0, the standard mask of class c is 255.255.255.0. Some of the bits of the last octet of mask can be used to create

subnets. If we take 3 high order bits of the last octet i.e. from host id portion, we get a subnet mask of 255.255.255.224. Let us see the binary representation of this:

Address: 11001101 .00010001 .00001101 . 00000000

Mask: 11111111 .11111111 .11111111 .11100000

The underlined three bits in the mask are used for subnetting and remaining five for hosts in specific subnet. With three bits 2^3 i.e. 8 subnets can be created each having host ids 2^5 i.e. 32. The first and last id of any subnet is kept reserved and is not allotted to any device. So in this example following subnets can be created each with subnet mask of 255 .255 .255 .224 :

Subnet No.	Subnet Network Id	Range of Host Ids
1	205 .17 .13 . 0	205.17.13.1 to 205.17.13.30
2	205 .17 .13 . 32	205.17.13.33 to 205.17.13.62
3	205 .17 .13 . 64	205.17.13.65 to 205.17.13.94
4	205 .17 .13 . 96	205.17.13.97 to 205.17.13.126
5	205 .17 .13 . 128	205.17.13.129 to 205.17.13.158
6	205 .17 .13 . 160	205.17.13.161 to 205.17.13.190
7	205 .17 .13 . 192	205.17.13.193 to 205.17.13.222
8	205 .17 .13 . 224	205.17.13.225 to 205.17.13.254

Mask of 255 .255 .255 . 224 can also be denoted as / 27 as 27 bits are being used for sub networking. In CIDR notation it can be represented as 205. 17. 13. 32/ 27

4.6 ROUTING ALGORITHMS

Routing is a technique of forwarding packets from sender to receiver. Routing algorithm is a technique to decide the route or path to transfer data packets from sender to the receiver. It is part of network layer software that decides the output line for the incoming packet. For datagrams this decision is made for every packet but for virtual circuit the decision is made only once and path is fixed. Routing algorithm computes the best path for transmission which is a least cost path.

The goals of routing are:

- **Simplicity:** Routing algorithm should be simple with minimum overheads.
- **Robustness:** The algorithm must perform correctly even during hardware failures or during high load in the network.
- **Stability:** It should be stable to handle any issues in the network. Also the convergence should be fast.
- **Flexibility:** Routing algorithm should adapt to changes due to network bandwidth, delay or failure of any links.
- **Fairness:** The algorithm must provide equal opportunity to all the nodes to transmit packets according to quality of service requirements.

- **Optimality:** The algorithm should be able to find the best route depending on the metrics like throughput, delay etc.

4.6.1 Types of Routing Algorithms

Routing Algorithms can be broadly categorized into non-adaptive and adaptive routing algorithms.

Non-adaptive algorithms do not make their routing decisions on any estimates of the traffic or the current network topology. Choice of the route from router A to router B is computed in advance and in offline mode. The same information is downloaded by routers when the network is booted. It is a static routing and does not respond to failures.

Adaptive algorithms change their routing decision based on the topology and traffic. It is a kind of dynamic routing algorithm.

4.6.2 Non-adaptive Algorithms

The static routing tables are constructed for packet transmission and router stores these routing tables when the network is up. Once the static paths are available with the routers in the network the packet transmission starts. The change in topology or traffic conditions do not have any effect on the previously constructed routing tables. There are majorly two types of non-adaptive algorithms.

4.6.2.1 Flooding

In a simple technique of flooding when the data packet arrives at any router it is transmitted to all the outgoing links except the one from which it arrived on. Each router make decisions based on the local knowledge and not with the complete picture of the network.

Large number of duplicate packets are generated in flooding if it is uncontrolled.

4.6.2.1.1 Controlled flooding

To control the flooding, hop counter is added in the header of the packet, it is initialized to the length of the path from sender to receiver. Hop counter is decremented at each hop and packet is discarded if hop counter reaches to zero. If the length of path is unknown the hop counter is put as the full diameter of the network. In this technique lot of duplicate packets will be generated if the hop count is large.

Sequence Number Controlled Flooding: Another technique to control the flooding is to avoid transmitting them again by keeping the track of packets at router. A sequence number is put by the source router for every transmitted packet. Each router maintains a list of the sequence numbers of packets transmitted by it per source. If the incoming packet to the router is in the list then it is not flooded.

Reverse Path Forwarding: It works on the technique that any broadcasted packet arriving at the router is checked by the router to see if the packet has arrived on the link that is generally used for transmitting packets towards sender for broadcast. If it is so, then it is assumed that broadcasted packet followed the best route from the router and is the first copy of the packet that has arrived to the router. If the packet has

arrived from any other link apart from that, it is considered as duplicate and is discarded. It is an efficient technique and is simple to implement.

4.6.2.1.2 Selective Flooding

Some applications like online games, do not require the packets to be delivered to all but only to the selected group. In this technique, packets are transmitted only to a specific group. It is also known as multicast routing. Group is created of the routers, each group is identified by multicast address and routers know to which group they belong. It is done by the techniques known as multicast spanning tree.

Flooding has the following advantages:

- It ensures that packet is delivered to every node which is effective for broadcasting.
- It is robust. Even if several routers go down, the path from sender to receiver will be available.
- In flooding router needs to know its neighbours so it can be used as basic technique for other routing algorithms as well.
- It chooses the shortest possible path for transmission.

4.6.2.2 Random Walk

In this algorithm, a packet is sent by node to one of its neighbours randomly. It suffers from the drawbacks of dead-end when no out-link is there. Also, when there is no route to another network then it becomes kind of infinite cycle or a spider-trap.

4.6.3 Adaptive Algorithms

The routing decisions in this category of algorithms changes due to changes in topology and network traffic. These are the dynamic algorithms, the router get the information either from the adjacent routers or from all the routers. These algorithms differ in the way of getting network information, time of change of routes, metrics for deciding optimal routes.

The adaptive algorithms are categorized into centralized, isolated and distributed algorithms.

4.6.3.1 Centralized Algorithm

As the name suggests, a central node has the information of entire network and it takes the routing decisions. It has an advantage that the resources are required only for the central node to store and process complete data but on the other hand it has a disadvantage that in case of failure of central node the entire network will go down.

4.6.3.2 Isolated Algorithm

In this algorithm the node takes a decision in isolation i.e. without seeking information from other nodes. The status of the link to which packet is transmitted is not known to the node. It has a major disadvantage of delay in transmission in case of congestion in the link. Examples of this type of algorithm are:

- **Hot Potato:** In this algorithm as and when packet arrives at node, it tries to transmit it to the shortest output queue, without considering whether the link is correct link to the destination or not.
- **Backward Learning:** In this algorithm, routing tables at each node is modified based on the information received through incoming packets. It is generally implemented by adding identity of source node together with hop counter. At the node, number of hops the packet has taken to reach to the node is noted. If the current hop count is better than the previously stored value in the node, it is replace with the new value. This algorithm does not store the previous records, so in case the best route is down, it does not have the information about second best route. The process is started all over again.

4.6.3.3 Distributed Algorithm

It is a decentralized algorithm, every node receives information from the neighbouring node and takes its own decision. The least-cost path is calculated in a distributed manner. It has an advantage that each node can dynamically change the decision and choose the best path for transmitting packets. But, this approach may result in delay due to time required by each node to take decision.

4.6.3.4 Distance Vector and Link State Packet Routing Algorithms

The two main algorithms under category of adaptive algorithms are distance vector routing and link state packet routing.

4.6.3.4.1 Distance Vector Routing

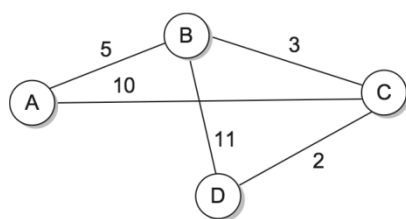
It is a dynamic routing algorithm in which each router maintains a table containing information about the best route to each destination and the link for transmitting packet to the destination. Routers exchange these tables with the neighbours and accordingly every router gets the information about best link to reach to the destination.

Every router maintains routing table containing one entry corresponding to each router in the network. Every table entry has two fields, one for preferred outgoing link to reach to destination and distance to the destination either in terms of hops or any other metric. The routing table is shared with all the neighbours and they update their tables accordingly.

Fig. 4 below shows the distance calculation using this algorithm. Every router in the network has its routing table specifying the distance to reach to another router. The metric used for calculating distance is the delay. Consider a case where distance from router J to G has to be calculated. The neighbours of J have its own routing table to reach to any other router as specified in figure.

Router J is also maintaining its routing table but based on the information from neighbours it will update its routing table which is explained below.

The information maintained currently by J to reach to its neighbours are:



Link Packets by

A	B	C	D
B 5	C 3	A 10	B 11
C 10	D 11	B 3	C 2
		D 2	

Fig. 5 Link State Packets formed by routers

The link packets prepared by routers are shared with all and accordingly routers make their own routing table to reach to the destination. Sequence number of link state packets and age of packets help in rejecting the obsolete packets.

4.6.4 Comparison of Distance Vector Routing and Link State Routing

Based on the key points in the algorithms these two major adaptive algorithms can be compared as below:

Key Points	Distance Vector Routing	Link State Packet Routing
Underlying algorithm	Bellman Ford Algorithm	Dijkstra Algorithm
Metric for distance Calculation	Hop Count	Link Cost
Information Sharing	Router shares the information to its neighbours about its distance from every other router in the network	Router shares the information to every router about its distance from its neighbours
Updates in the tables	Frequent	Event triggered
Convergence	Slow	Fast
Issues	Count to infinity (routing loop) and creates more traffic as compared to link state	High processing requirements and memory requirements

4.7 SUMMARY

This module covers the functionality of network layer. The main function of the network layer is to provide end-to end delivery of packets between source to destination. It provides the routing of the packets in multi-hop networks. The two approaches of packet transmission – virtual circuit which is connection oriented and datagram which is connectionless service have been discussed. As the network layer deals with the packet transmission in the network, so unique identification or the addressing of nodes is required. IP addressing is the technique

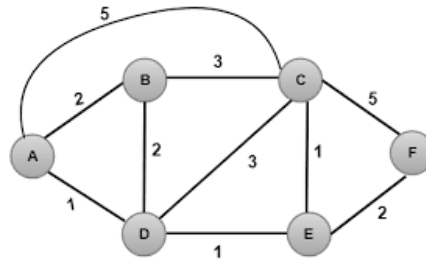
used for uniquely identifying nodes in the network. Two versions IPv4 and IPv6 are used for this identification. IPv4 though most commonly used has a limitation to uniquely identify only 2^{32} nodes as it is a 32 bit address. With the increase of number of nodes in the internet addressing has been moved towards IPv6 which is 128 bit addressing scheme.

The earlier classful addressing for the networks is not very beneficial for the organization that require limited number of IP addresses. If the organization has requirement of 120 IP address then getting one Class C network address, which can address 256 nodes will be a waste of IP addresses as remaining 136 (256-120) addresses will not be ever used by the organization but at the same time it cannot be allocated to any other organization. Considering this subnetting was introduced, through which any number of IP addresses required by the organization was allocated thus avoiding unused IP addresses.

Routing another major function of network layer has underlying algorithms, the adaptive algorithms which are dynamic in nature are preferred inspite of the high complexity than static or non-adaptive algorithm. Distance vector routing and link state routing are examples of adaptive algorithms being used for routing. Each of these two has its advantages and disadvantages. Routing information protocol (RIP) and interior gateway routing protocol (IGRP) are examples of distance vector routing whereas an interior gateway routing protocols open shortest path first (OSPF) and intermediate-system to intermediate-system (IS-IS) are examples of link state routing algorithm.

4.8 REVIEW QUESTIONS

- a. List the functions of network layer and the services it provides to upper layers.
- b. Which are the factors that make datagram service better than the virtual circuit? Also, give the example of datagram service.
- c. What was the requirement for IPv6 and how many unique addresses it can assign to the devices connected to internet?
- d. Convert these IP addresses into decimal notation and identify in which class these addresses lie.
 - 00011111.11110000.00110011.00000100
 - 10111111.00110000.00000100.00110000
 - 11000001.01111111.11111100.11110000
 - 11111110.00001100.11111111.00001110
- e. How the subnetting helps in improving network performance? What does notation 192.68.12.0/27 signifies.
- f. Determine the subnetwork address for 172.16.18.19/20.
- g. Consider class C network with address 204.18.4.0, with subnet mask of 255.255.255.240, identify the subnetwork ids and host range in each subnet.
- h. Enumerate the differences between adaptive and non-adaptive routing algorithms.
- i. Explain how flooding helps in broadcasting. What are disadvantages of flooding?
- j. In the below figure, calculate the shortest path from A to F by showing all complete process of calculating distances.



REFERENCES & FURTHER READING

- Data Communication & Networking by Behrouz Forouzan
- Computer Networks by Andrew Tannenbaum
- Data and computer Communications by William Stallings
- Computer Networking: A Top-Down Approach by James Kurose and K.W. Ross

UNIT 5: TRANSPORT LAYER

STRUCTURE

- 5.0 Objectives**
- 5.1 Introduction**
- 5.2 Transport Layer's Services**
- 5.3 Elements of Transport Layer Protocols**
 - 5.3.1 Addressing**
 - 5.3.2 TCP Connection Establishment**
 - 5.3.3 TCP Connection Termination**
- 5.4 Flow Control and Buffering**
- 5.5 Transport Layer Protocols**
 - 5.5.1 Introduction to UDP**
 - 5.5.2 UDP Segment Structure**
 - 5.5.3 Introduction to TCP**
 - 5.5.4 TCP Segment Structure**
- 5.6 Review Questions**
- 5.7 Summary**
- 5.8 References for Further Readings**

5.0 OBJECTIVES

At the end of this unit, you will be able to:

- Understand the issues related with transport layer.
- Discuss different services of transport layer.
- Discuss in detail the various elements of transport layer protocols.
- Understand the concept flow control and buffering.
- Discuss frame formats of TCP and UDP.
- Discuss steps for connection establishment and connection release.

5.1 INTRODUCTION

The transport layer is the OSI model's link pin. It is not just another layer but the heart of the whole protocol hierarchy. Protocols on this layer monitor the transmission on another device of data from one application programme into an application programme. This is the first OSI end-to-end layer. It provides services on the upper layer to use the network layer services and other protocols on the lower layer.

We should be aware that the Internet is made up of many physical networks such as LANs, MANs, and WANs that are linked together using a variety of media (wired and wireless) to enable data to be transferred from one network to another. The data in a transmission may be transformed as it travels from one network to another, i.e., it may be encapsulated in various forms and lengths of packets. The complexity of the physical networks at the transport layer is hidden from upper-layer protocols. Individual physical networks appear to the upper layers as a simple homogeneous network capable of efficiently transporting data. And if an Ethernet is replaced by an FDDI (Fiber Distributed Data Interface) in the LAN portion of the internet, the upper layers are unaffected. For them, the Internet is a single, mostly static network. This is provided by the transport layer.

Examples of transport layer protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

5.2 Transport Layer's Services

Let's start with the services that a transport protocol can or should provide to upper layer protocols. The world for transportation services is depicted in below Figure. A transport object, which may be an application mechanism like http, telnet, etc., offers services to upper layer users. Depending on the network layer's facilities, this local transport entity interacts with any remote transport entity. As stated, a transport protocol's general service is to provide end-to-end data transport while shielding the upper layer user from the specifics of the underlying network infrastructure.

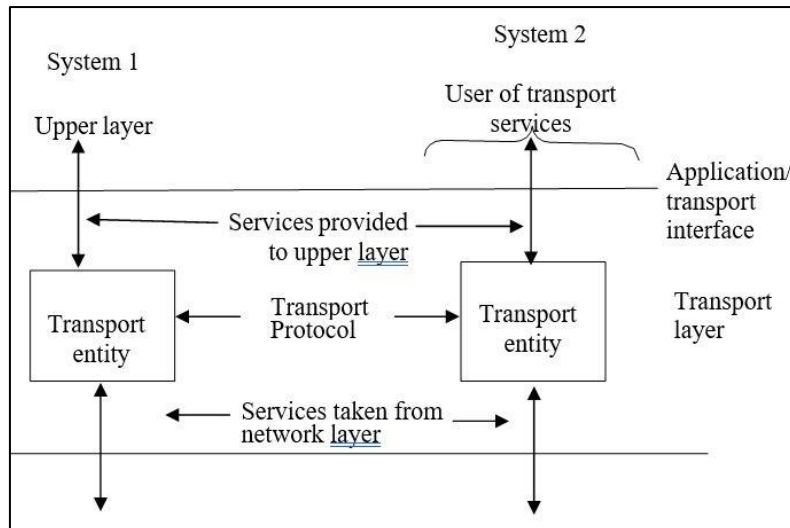


Figure: Transport Entity Environment

The services offered by the transport layers are typically classified into the following categories.

Types of services

1. Quality of service
2. Data transfer
3. Connection management
4. Expedited delivery

The Internet provides two basic types of services:

(i) Connection-oriented services and (ii) Connectionless, or datagram services.

Connection-Oriented

The most of protocol service is a connection-oriented service. It allows users to create, maintain, and terminate a logical link between them. Handshake procedures are another name for this. TCP is used to provide Internet connection-oriented services (Transmission Control Protocol). TCP provides a variety of services to an application, including reliable transportation, flow management, and congestion control. These programmes have been recognised. To preserve throughput, the TCP's congestion management function is used.

Connectionless Service

In connectionless service, before sending the packet, there is no handshaking. There are still no systems in place to manage traffic flow and congestion. Data can be transmitted more quickly since there is no handshaking protocol. However, although these services are acknowledged, there is no secure data transfer. UDP is the name of the Internet's connectionless operation (User Datagram Protocol). Online telephony and video conferencing are two examples of connectionless service applications.

1. Quality of Service

The upper layer protocol users should be able to define the types of transmission service quality that will be given by the transport protocol entity. Based on these requirements, the transport agency tries to make the best use of the underlying link, network, and other resources in order to provide the collectively requested services to the best of its capacity. However, these networks are constrained by the network layer services' internet capabilities.

Examples of services that might be requested are:

- Expedited delivery of packet
- Acceptable error and loss levels of packet
- Desired average and maximum delay in transmission of a packet
- Desired average and minimum throughput
- Priority levels.

You are aware that IP is a network layer standard protocol. IP does include a priority parameter, as well as binary specifications for normal or low latency, normal or high throughput, and normal or high reliability. As a result, the transport entity will communicate with the internetwork entity. To achieve desired throughput, the network can change flow control parameters and the amount of network resources allocated to a virtual circuit.

You must be aware that different applications may require different quality of services. For example:

- A file transfer protocol might require high throughput. It may also require high reliability to avoid retransmissions at the file transfer level.
- A transaction protocol (e.g., web browser-web server) may require low delay.
- An electronic mail protocol may require multiple priority levels.

The inclusion of a quality-of-service facility within the protocol is one approach to providing a variety of service qualities; transport protocols typically follow the same approach. Another option is to have different transport protocols for different types of traffic; the ISO-standard family of transport protocols takes this approach.

2. Data Transfer

The major purpose of a transport protocol is to transfer data between two transport entities. User data and control data must be transferred from one end to the other on the same channel or on separate channels. Full-duplex service is required. Half-duplex and simplex modes may also be available to accommodate the needs of specific transport users.

If the network layer performs a similar function, why is it necessary at the transport layer? The network layer manages the hop-by-hop delivery of individual packets but sees no

relationship between them, even if they are part of a single message. Each packet is treated as a separate entity. The transport layer, on the other hand, ensures that not just one packet but the entire sequence of packets is delivered.

3. Connection Management

When providing connection-oriented service, the transport entity is in charge of establishing and terminating connections. It is possible to provide both symmetric and asymmetric procedures for connecting establishments.

Connections can be terminated abruptly or gracefully. Data in transmission may be lost if the connection is abruptly terminated. A graceful termination prevents either party from terminating until all data has been delivered.

4. Expedited Delivery

The expedited delivery of data is a service similar to that provided by priority classes. Some data submitted to the transportation service may supersede previously submitted data. The transport entity will make every effort to have the transmission facility send the data as quickly as possible. The transport entity will interrupt the transport service user at the receiving end to notify it of the receipt of urgent data. Thus, the expedited data service is an interrupted mechanism that is used to transfer occasional urgent data, such as a terminal break character or an alarm condition. A priority service, on the other hand, may devote resources and adjust parameters in such a way that, In contrast, a priority service might dedicate resources and adjust parameters such that, on average, higher priority data are delivered more quickly.

5.3 ELEMENTS OF TRANSPORT LAYER PROTOCOLS

Transport layer protocols such as TCP and UDP are used to implement the services provided by the transport layer. We'll go over them in the next unit. In this section, we will look at some of the most important components of the transport layer. A transport layer is in charge of hop-by-hop processes.

5.3.1 Addressing

The issue at hand is simply this: how will a user process establish a connection to a remote user process? How should the address of a remote process be specified? The most common method is to specify a transport address to which the process can listen for connection requests. These end points are known as ports or TSAPs in Internet jargon (Transport Services Access Points). Similarly, these network layer end points are referred to as NSAPs (Network Services Access Points). The diagram below depicts the connection between a user process as host 1 and a remote process (server) as host 2.

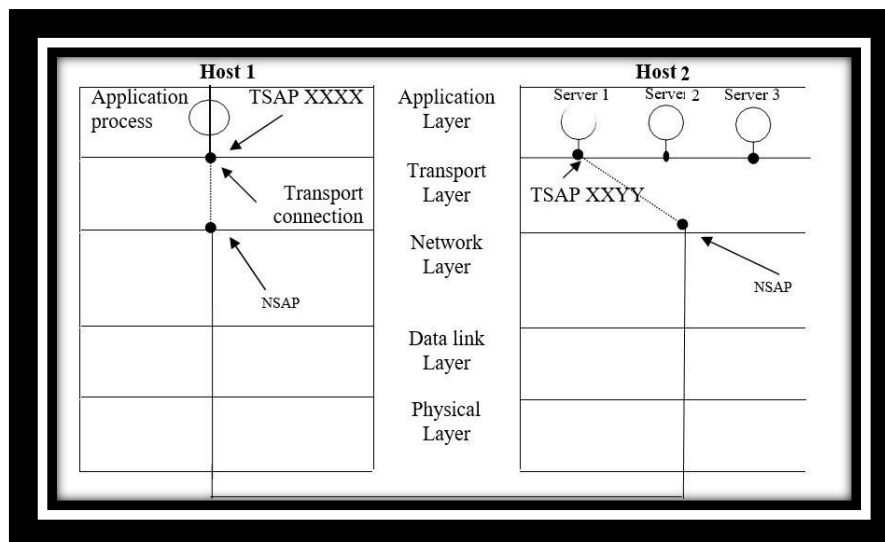


Figure: Transport Connections between a Client and a Server

The following steps may be needed for establishing a transport connection:

Assume that an application process running on host-1 wants to know the weather forecast for the next day, so it sends a **CONNECT** request specifying its **transport connection point number** TSAP XXXX as well as the weather forecasting server's TSAP number XXYY, which will establish a transport connection with the destination. This action results in a transport connection being established between the application process on host 1 and a server 1 on host 2.

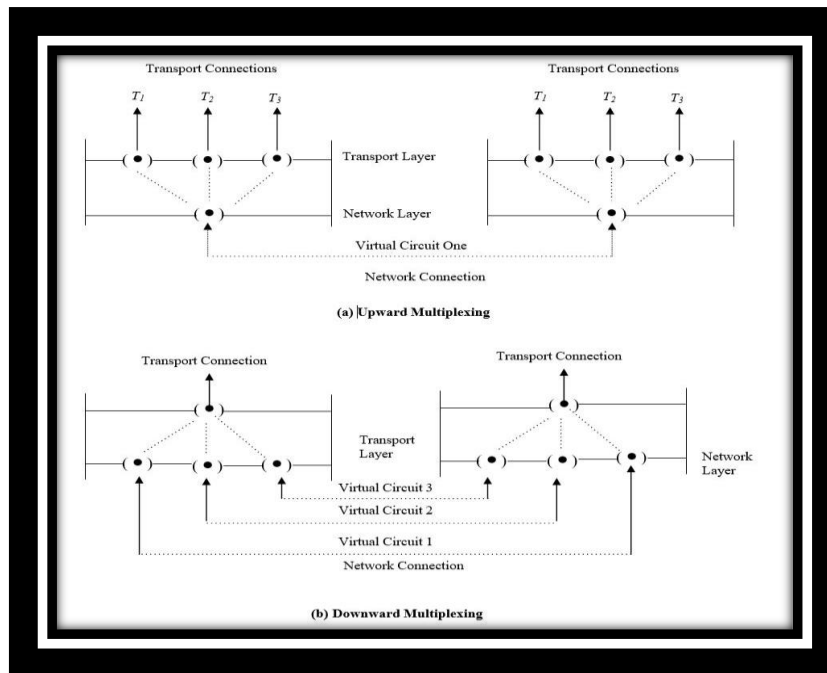
The following methods are used to know that the address of the destination server process is attached to a particular transport connection by the host user

Initial connection protocol

A process server acts as a proxy server for less heavily used servers in the scheme. Instead of each server listening at a well-known port address for a connection request, a process server listens to groups of ports at the same time and notifies a specific server to act upon receiving a connecting request to perform the required work. The new server then handles the request, while the process server returns to waiting for new requests. As a result, a rarely used server should not be listening to ports all of the time via a process server.

Some commonly used services are assigned "well-known address" (for example, time sharing and word processing).

Multiplexing: The transport layer can also perform multiplexing. Upward multiplexing, defined as the multiplexing of multiple transport connections on a single network link, and downward multiplexing, defined as the splitting of a single transport connection among multiple lower-level connections, are the two forms of multiplexing techniques. Figure below shows how this works.



Downward multiplexing or splitting could be used to provide more bandwidth than can be managed by a single virtual circuit. One way out of this is to open several network connections and round robin traffic, as shown in the above figure (b). The effective bandwidth is increased by a factor of k when the network connections are opened.

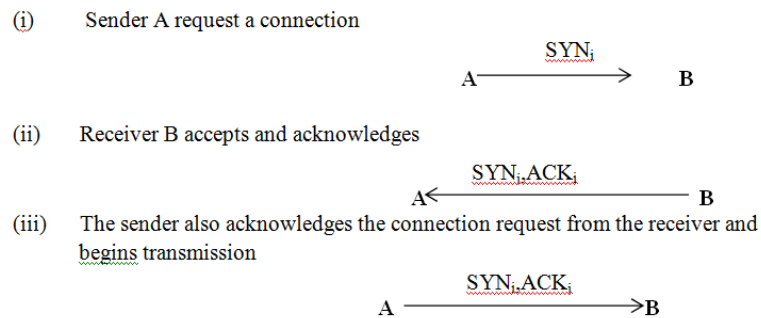
5.3.2 TCP Connection Establishment

A link must be formed before data can be transmitted. Connection establishment must account for a network service's unreliability, which can result in the loss of ACK, data, and SYN. As a result, these packets would need to be retransmitted. Remember that establishing a relation necessitates the exchange of SYNs.

Steps of typical three-way handshake operations are:-

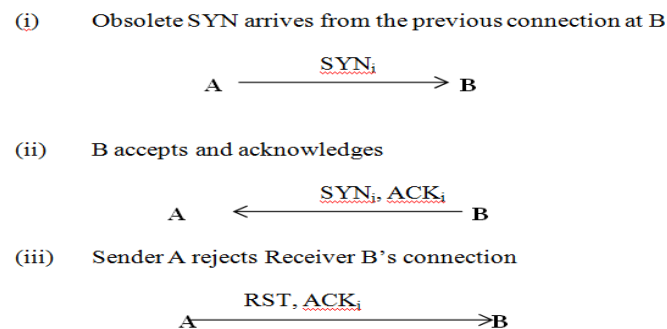
- 1) At the sender side transport entity A, initiates the connection to transport to entity B by setting SYN bit.
- 2) At the sender side transport entity B, acknowledges the request and also initiates a connection request.
- 3) An acknowledges to connection request of A and B sends a retransmission and B retransmits.

Normal Transmission



Delayed arrived of SYN packet:

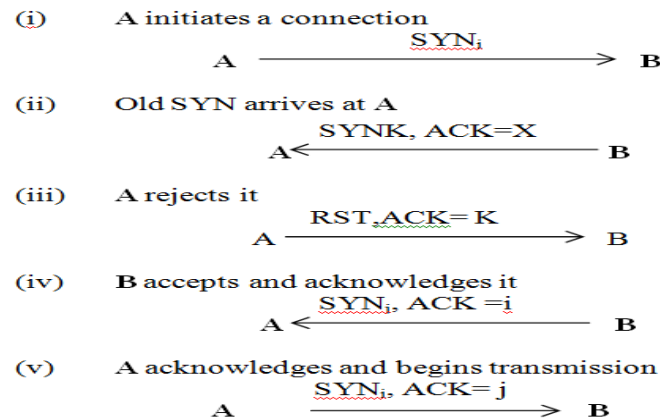
Now, let us test how this mechanism handles delayed SYN and ACK packets. It is shown that old SYN X arrives at B after the close of the relevant connection as shown below. B assumes that this is a fresh request and responds with $\text{SYN}_j, \text{ACK}_i$. When A receives this message, it realizes that it has not requested a connection and therefore, sends an RST, ACK_j . Note that the ACK_j portion of the RST message is essential so that an old duplicate RST does not abort a legitimate connection establishment.



Delayed SYN and ACK

The final example shows a case in which an old SYN, ACK arrives in the middle of a new connection establishment. Because of the use of sequence numbers in the acknowledgements, this event causes no harm.

Both transport entities must agree on their initial sequence number, which must be different, as part of the three-way handshake protocol. What is the aim of having different sequence numbers?



5.3.3 TCP Connection Termination

TCP uses a method close to that of relation establishment. TCP allows for a graceful close that requires the closure of each link path independently. When an application informs TCP that it has no more data to send, a termination occurs. To be accepted, each side must clearly recognise the other's FIN. A graceful close occurs successfully with the following moves.

The sender must send FIN_j and receive an ACK_j

- It must receive a FIN_j and send an ACK_j
- It must wait for an interval of time, to twice the maximum expected segment lifetime.

Acknowledgement Policy:

When a data segment arrives that is in sequence, the receiving TCP entity has two options concerning the timing of acknowledgment:

- **Immediate:** When data are accepted, immediately transmit an empty (no data) segment containing the appropriate acknowledgment number.
- **Cumulative:** When data are accepted, record the need for acknowledgment, but wait for an outbound segment with data on which to piggyback the acknowledgement. To avoid a long delay, set a window timer. If the timer expires before an acknowledgement is sent, transmit an empty segment with the appropriate acknowledgement number.

5.4 Flow Control And Buffering

In order to support a flux control mechanism, similarities and differences exist between the data link layer and the transport layer. Whereas flow control is a relatively simple mechanism at the data link layer, it is a rather complex mechanism at the transport layer, for two main reasons:

- Flow control at the transport level involves the interaction of three components: TS users, transport entities, and the network service.
- The transmission delay between transport entities is variable and generally long compared to actual transmission time.

The following delay may arise during the interaction of the two transport entities A and B:

- (i) Waiting time to get permission from its own transport entity (interface flow control).
- (ii) Waiting time by A to have permission to send the data to B.
- (iii) Waiting time as network layer services.

In any case, once the transport entity has accepted the data, it sends out a segment. Later on, it receives an acknowledgement that the data has been received at the remote end. It then sends a confirmation to the sender, about the same.

On the others side, in case of flow control problem, firstly we present two ways of coping with the flow control requirement by using a fixed sliding-window protocol and a credit scheme.

The performance of the transport connection can be reduced in long-term cases by a conservative flow management scheme. By optimistically granting credit for space, the recipient could potentially increase the performance. The buffer allocation scheme is called dynamic. E.g. a recipient's buffer is full but expects to space for two segments in a time of roundtrip propagation, a loan of 2 can be sent immediately. This scheme can increase outcomes and do no harm if the receiver can keep up with the sender. However, if the sender is faster than the receiver, segments that require a re-transmission may be discarded. Because retransmissions are not otherwise necessary with a reliable network service, an optimistic flow control scheme will complicate the protocol.

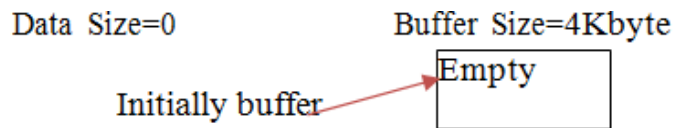
The optimal compromise between source and target buffer depends on the type of traffic that the connection performs. It is better not to dedicate buffers for low bandwidth, bursty traffic such as those produced by an interactive terminal, but to acquire buffers in both directions. The recipient must keep a copy of the TPDU (Transport Protocol Data Unit) until acknowledged since the sender cannot be certain that the recipient will be able to acquire a buffer. On the other hand, it is better to allow data to flow at high speed for file transfer and other high bandwidth traffic if the recipient has a whole buffer window. Therefore, for low bandwidth explosive traffic, the buffer on the sender is better, and a buffer on the recipient is better for high bandwidth smooth traffic.

TCP Flow Control

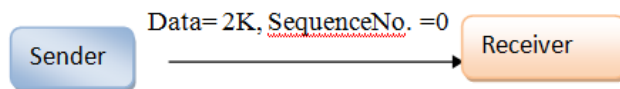
The method of controlling traffic between two end points is known as flow control, and it is used to prevent the sender from sending more data as compare to the recipient capacity to process it. To prevent the sender from overflowing, TCP offers a flow control service. TCP handles flow control at the receiver's buffer by using a sliding window with a credit scheme. The scheme allows the receiver to have more control over the data flow. A part of a credit scheme can be recognized without the guarantee of new credit, and vice versa. The two are intertwined in a fixed sliding window control (used at the data link layer).

E.g. assume that the sender wants to send application data to the receiver.

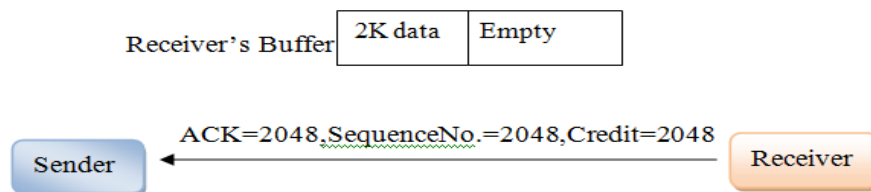
i) The receiver has 4 K byte buffer which is empty as shown below:



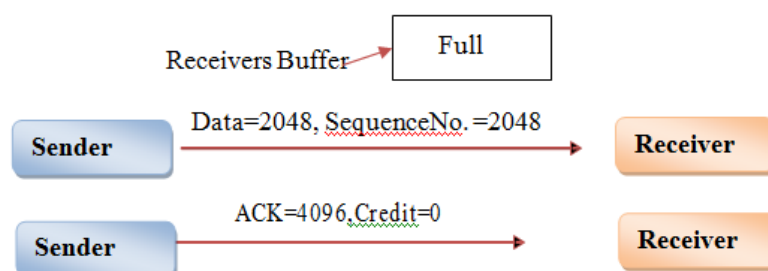
ii) Sender transmits 2 K byte segment (data) with sequence number 0 as shown below:



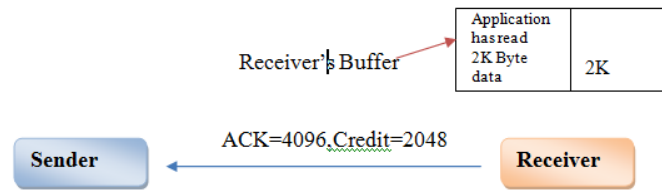
iii) The packet is examined at the receiver, after that it will be acknowledged by it. It will also specify the credit value (window size) of the receiver. Until the application process running at the receiver side removes some data from buffer, its buffer size remains fixed at 2048. Therefore, credit value (window size) is 2048 byte.



iv) Now the sender transmits another 2048 bytes, which is acknowledged, but now the advertised window (credit) is 0.



v) The sender must stop sending data until the application process on the receiving host has removed some data from the buffer, at which time TCP can advertise, a large window (credit value).



When the credit size is zero, normally there is no transmission from the sender side except in two situations:

- a). Urgent data requires to be sent
- b). Sender wants to know the credit size and the next byte expected by the receiver.

Both senders and receivers can delay transmission to save money on their end. If a sender knows that a receiver window's buffer capacity is 8 kilobytes and it has only received 2 kilobytes, it can buffer it at the sender side before it receives more data from the application phase. Similarly, if the receiver has to send any data to the sender, the acknowledgment can be delayed until the data is ready, and the acknowledgement can then be piggybacked. As a result, the primary motivation for delaying acknowledgment is to save bandwidth.

5.5 TRANSPORT LAYER PROTOCOLS

The protocol layer just above the Internet Layer is the *Host-to-Host Transport Layer*. This name is usually known to *Transport Layer*. The two most important protocols in the Transport Layer are *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP). TCP provides reliable data delivery service with end-to-end error detection and correction. UDP provides low-overhead, connectionless datagram delivery service. Both protocols deliver data between the Application Layer and the Internet Layer. Applications programmers can choose whichever service is more appropriate for their specific applications.

5.5.1 Introduction to UDP

As, you know that UDP and the transport protocol are unreliable. Apart from multiplexing, UDP adds little to the IP protocol and some error correction.

If we choose UDP for application development instead of TCP, then the application speaks nearly directly to the IP layer. For the multiplexing/demultiplexing services, UDP takes messages from the application process, attaches the source and destination port numbers to the fields, adds two small fields and passes the resulting segment on to the network layer. The network layer encapsulates the segment into an IP datagram and makes best efforts for transferring the segment to the receiving host without making a swipe between sending and receiving transmission layer entities. . If the segment arrives at the receiving host, UDP uses the destination port number to deliver the segment's data to the desired application process.

DNS and many other applications are better suited for UDP for the following reasons:

No connection establishment: Since UDP does not delay the establishment of a connection, hence UDP-DNS is faster as compared to TCP. But, HTTP uses TCP instead of UDP because reliability in text Web pages is critical.

More support for the client: In the end systems, TCP maintains the connection state. This connection state includes the reception and sending of buffers, congestion control parameters and parameters for sequence and number recognition. This state information is necessary for the implementation and congestion control of reliable data transfer service by TCP. On the other hand, UDP has no connecting status and does not follow any of these parameters. When the application runs via UDP instead of TCP, a server dedicated to a particular application typically supports more active clients. Due to the absence of reliable data service and congestion control, UDP does not have to maintain the receipt or send of buffers (connection state, congestion control parameters, sequence and recognition number configurations), and therefore does not have to monitor the sending and receiving of buffers.

Small packet header overhead: The TCP segment has 20 bytes of header overhead in every segment, whereas UDP has only eight bytes of overhead.

Can a reliable UDP application be developed? Yes, it can be done at the application level through the addition of recognition and retransmission mechanisms. Many proprietary streaming applications today are only possible this is they run over UDP, but they have built acknowledgements and retransmissions into the application in order to reduce packet loss.

5.5.2 UDP Segment Structure

UDP is an end to end transport level protocol that adds only port addresses, checksum error control and length information to the data from the upper layer.

The data for the application covers the UDP segment data field. For example, for DNS, a query or response message is contained in the data field. Audio samples fill in the data field for a streaming audio application. A user datagram is called the packet generated by UDP.

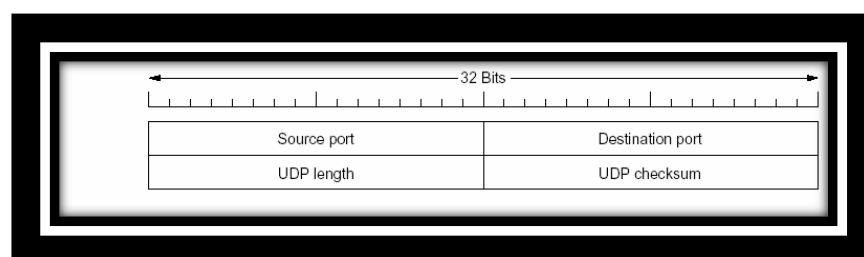


Figure: UDP segment structure

The UDP header has only four fields, each consisting of two bytes. Let us discuss each field separately:

Source port address:

It is the address of the application program that has created the message.

Destination port address:

It is the address of the application program that will receive the message.

Total length:

It specifies the length of UDP segment including the header in bytes.

Checksum:

The checksum is used by the receiving host to check whether errors have been introduced into the segment. In truth, the checksum is also calculated over a few of the fields in the IP header in addition to the UDP segment.

5.5.3 Introduction to TCP

TCP is a protocol that allows two processes (TCP users) to communicate reliably over a range of networks and Internets, both stable and unreliable. TCP is a Connection Oriented protocol. Any link is treated as if it were a stream of bytes. In contrast to UDP, the user application does not need to bundle data in individual datagrams. A connection between the sender and the receiver is needed in TCP. TCP requires a VC (Virtual Circuit) at the IP layers to establish this connection, which must be active for the duration of the transmission. The data is placed in allocated buffers and sent in segments via TCP. In addition, TCP provides two useful facilities for labeling data; **push (PSH)** and **urgent (URG)**.

When the PSH (data push) bit is set, this is an indication that the receiver should pass the data to the upper layer immediately. The URG (Urgent) bit is used to indicate that there is data in this segment that the sending side upper layer entity has marked as urgent. The location of the last byte of this urgent data is indicated by the 16 bit urgent data pointer field.

Both IP and UDP treat multiple datagrams belonging to a single transmission as entirely separate units, un-related to each other. TCP on the other hand is responsible for the reliable delivery of entire segments. Every segment must be received and acknowledged before the VC is terminated.

5.5.4 TCP Segment Structure

TCP uses only a single type of protocol data unit, called a TCP segment. The header is shown in Figure below. Because one header must perform all protocol mechanisms, it is rather large, with a minimum length of 20 octets. A segment beginning with 9 fixed format 20 byte headers may be followed by header option. After the options, if any, up to $65,535 - 20$ (IP header) – (TCP header) = 65,445 data bytes may follow. A Segment with no data is used, for controlling messages and acknowledgements.

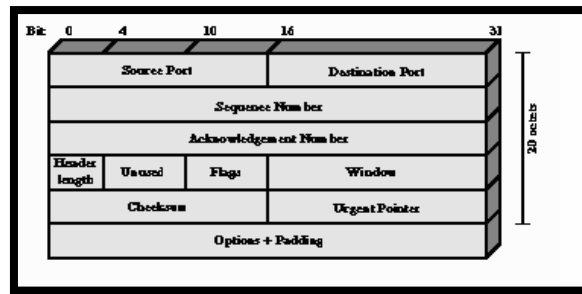


Figure: TCP header format

Source port (16 bits):

Source service access points and identify local end points of connection.

Sequence number (32 bits):

Sequence number of the first data octet in this segment except when SYN is present. If SYN is present, it is the initial sequence number (ISN), and the first data octet is ISN + 1.

Acknowledgement number (32 bits):

A piggybacked acknowledgement contains the sequence number of the next byte that the TCP entity expects to receive and not for last byte correctly received. Separate number field and acknowledgement number fields are used by the TCP sender and the receiver has to implement a reliable data service.

Flag:

Since each flag is one bit long, and as there are 6 flags: **URG**ent Pointer, **ACK**nowledgement, **P**ush(**PSH**), **R**eset (**RST**), **SYN**chronisation and **FIN**ish. This makes the Flags section 6 bits in total. The most popular flags are the “SYN”, “ACK” and “FIN”, used to establish connections, acknowledge successful segment transfers and, lastly, terminate connections.

URG: Used to indicate that there is data in this segment which sends the at the upper layer has marked urgent. The location of the last byte of this urgent data is indicated by the 16 bit urgent data pointer field.

ACK: Acknowledgement field indicates that the value carried in the ACK field is valid.

PSH: Push function. The receiver is represented to deliver the data to the application upon arrival, and not buffer it until a full buffer has been received.

RST: Reset the connection due to host crash or some other reason. **SYN:** Synchronise the sequence numbers.

FIN: No more data from sender.

Window (16 bits):

The number of data bytes that the receiver is willing to accept, starting with the one indicated in the acknowledgement field.

Checksum (16 bits)

It adds extra assurance. It calculates the checksums for the header, data, and conceptual pseudo header. The Checksum algorithm simply adds all of the 16-bit words in one's complement and then subtracts the total. As a result, the result should be 0 when the receiver performs calculations on the entire segment, including the checksum field.

Urgent Pointer (16 bits)

Points to the octet following the urgent data; this allows the receiver to know how much urgent data is coming.

Options (Variable):

At present, only one option is defined, which specifies the maximum segment size that will be accepted.

5.6 REVIEW QUESTIONS

1. Why User datagram protocol is called connectionless?
2. Explain the three types of address used by TCP/IP?
3. Why TCP services are also called stream delivery services?
4. When a process server acts as a proxy server?

5.7 SUMMARY

The two transport layer protocols (TCP and UDP) were thoroughly discussed in this unit. The transport port can be small (very simple) and provide minimal services. In such instances, the programme communicates directly with the IP. UDP is an example of a transport protocol that offers the bare minimum of features, including no power, link establishment, acknowledgment, or congestion handling. As a result, if the application is to be built around UDP, certain features must be supported inside the application. TCP, on the other hand, is a protocol that offers many benefits, including reliability, flow control, and link establishment to various application services. Nonetheless, the network layer protocol is often constrained by the services provided by the transport layer.

TCP link creation, UDP and TCP header formats, TCP flow control and finally congestion control were all discussed.

5.8 REFERENCES

1. A.S Tanenbaum, Computer Networks, 4th Edition, PHI, New Delhi
2. J.F. Kurose & K.W. Ross, Computer Networking, A top down approach featuring the Internet, Pearson Edition, New Delhi
3. William Stallings, Data and Computer Communication, Pearson Education, New Delhi
4. Behrouz Forouzan, Data communications and Networking, Tata Mc-Graw Hill.

UNIT 6: SESSION AND PRESENTATION LAYER

STRUCTURE

- 6.0 Objectives**
- 6.1 Introduction**
- 6.2 Concepts**
- 6.3 Session Layer Functions & Services**
- 6.4 Session Layer Design Issues**
 - 6.4.1 Remote Procedure Call (RPC)**
- 6.5 Presentation Layer**
 - 6.5.1 Functions: Presentation Layer**
 - 6.5.2 Design Issues: Presentation Layer**
- 6.6 Data Compression**
 - 6.6.1 Data Compression Methods**
- 6.7 Run Length Encoding**
- 6.8 Cryptography**
 - 6.8.1 Public Key Cryptography**
 - 6.8.2 Cryptography Features**
 - 6.8.3 Types of Cryptography**
 - 6.8.4 Cryptology Terminologies**
- 6.9 Review Questions**
- 6.10 Summary**
- 6.11 References and Further Readings**

6.0 OBJECTIVES

At the end of this unit, you will be able to:

- Understand the significance of session and presentation layer in data transmission.
- Understand the functions & services of session and presentation layer.
- Discuss in detail the various features of session layer and presentation
- .Explore different encoding, compression and encryption techniques.

6.1 INTRODUCTION

Session Layer is the 5th layer of the Open Systems Interconnection (OSI) reference model, which enables sessions between computers on a network to be established and terminated.. The session layer allows users on different machines to establish sessions between them. A session allows ordinary data transport, as does the transport layer, but it also provides enhanced services useful in some applications. A session might be used to allow a user to log into a remote timesharing system or to transfer a file between two machines.

The presentation layer is located at the sixth level of the OSI model; it is responsible for the delivery and formatting of information to the application layer for further processing or display. This type of service is needed because different computer architectures use different data representations. In this contrast, the presentation layer handles all issues related to data presentation and transport, including translation, encryption, and compression.

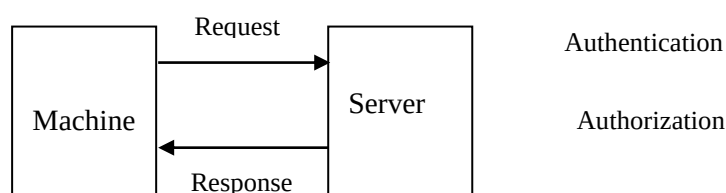
6.2 CONCEPTS

One of the services of the session layer is to manage dialogue control. Sessions can allow traffic to go in both directions (full duplex mode) at the same time, or in only one direction (half duplex mode) at a time. If traffic can only go one way at a time, the session layer can help keep track of whose turn it is through token management. For some protocols, it is essential that both sides do not attempt the same operation at the same time. To manage these activities, the session layer provides tokens that can be exchanged. Only the side holding the token may perform the data transmission.

Another session service is synchronization. Consider the problems that might occur when trying to do a 2-hour file transfer between two machines with a 1-hour mean time between crashes. After each transfer was aborted, the whole transfer would have to start over again and would probably fail again the next time as well. To eliminate this problem, the session layer provides a way to insert checkpoints into the data stream, so that after a crash, only the data transferred after the last checkpoint have to be repeated.

Hence responsibilities of session layer are:

1. Creating a session through **authentication**.g login into the remote system by using username and password.
2. The remote service responds through **authorization**.



3. The Session restoration through tokens and the same depends on type of server used. Whether server authorizes restoration on basis of token. The

restoration of session is not allowed in banking transactions. But same is allowed by email servers like gmail.

4. The session layer also helps in flow control and synchronization.
5. The creation of session depends on application interface instead of operating system used.

6.3 SESSION LAYER FUNCTIONS & SERVICES

The session layer provides the following services/functions:

1. Dialog Management: The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half- duplex (one way at a time) or full-duplex (two ways at a time) mode. Or can be said whose turn it is to talk.

In the ISO protocols, dialog management is implemented through the use of a data token. The token is sent back and forth, and a user may transmit only when it possesses the token.

2. Synchronization: The session layer allows a process to add checkpoints, or synchronization points, to a stream of data. For example, if a system is sending a file of 5000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 568, the only pages that need to be resent after system recovery are pages 501 to 568. Pages previous to 501 need not be retransmission.

The transport layer handles only communication errors, synchronization deals with upper layer errors. In a file transfer, for instance, the transport layer might deliver data correctly, but the application layer might be unable to write the file because the file system is full.

Users can split the data stream into pages, inserting synchronization points between each page. When an error occurs, the receiver can resynchronize the state of the session to a previous synchronization point. This requires that the sender hold data as long as may be needed.

Synchronization is achieved through the use of sequence numbers. The ISO protocols provide both major and minor synchronization points. When re-synchronizing, one can only go-back as far as the previous major synchronization point. In addition, major synchronization points are acknowledged through explicit messages. In contrast, minor synchronization points are just markers.

3. Activity Management: The session layer allows the user to delimit data in to logical units called activities. Each activity is independent of activities that come before and after it, and an activity can be processed on its own. Activities might be used to delimit files of a multi-file transfer.

Activities are also used for quarantining, collecting all the messages of a multi-message exchange together before processing them. The receiving application

would begin processing messages only after all the messages had arrived. This provides a way of helping insure that all or none of a set of operations is performed.

For example, a bank transaction may consist of locking a record, updating a value, and then unlocking the record. If an application processed the first operation, but never received the remaining operations (due to client or network failures), the record would remain locked forever.

6.4 SESSION LAYER DESIGN ISSUES

1. Establish sessions between machines

The establishment of session between machines is an important service provided by session layer. This session is responsible for creating a dialog between connected machines. The Session Layer provides mechanism for opening, closing and managing a session between end-user application processes, i.e. a semi-permanent dialogue. This session consists of requests and responses that occur between applications.

2. Enhanced Services

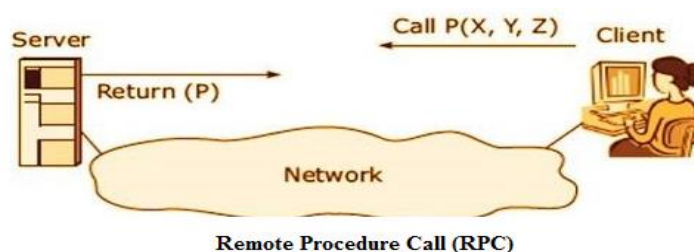
Certain services such as checkpoints and management of tokens are the key features of session layer and thus it becomes necessary to keep enhancing these features during the layer's design.

3. To help in Token management and Synchronization

The session layer plays an important role in preventing collision of several critical operation as well as ensuring better data transfer over network by establishing synchronization points at specific intervals. Thus it becomes highly important to ensure proper execution of these services.

Remote Procedure Call (RPC)

RPC is a protocol that one program use to request a service from a program located in another computer on a network without having to understand the network details. A procedure call is also called as function call or sub-routine. RPC uses client server model as shown below in figure.



It mainly includes five elements:

1. The client.
2. The client stub (stub: piece of code used for converting parameters)
3. The RPC runtime (RPC communication package).

4. The Server stub.
5. The Server.

The Client:

It is a user process which initiates RPC. The client makes a perfectly normal call that invokes a corresponding procedure in the client stub:

The Client Stub:

On receipt of a request it packs a requirement into a message and asks to RPC Runtime to send. On receipt of a result it unpacks the result and passes it to client.

RPC Runtime:

It handles transmission of messages between client and server.

The Server Stub:

It unpacks a call request and make a perfectly normal call to be invoke the appropriate in the server on receipt of result the procedure executes it and packs the result and ask the RPC runtime to send

The Server:

It executes an appropriate procedure and returns the result from a stub.

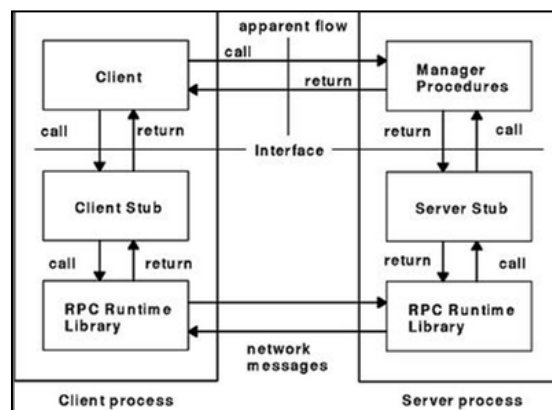


Figure: Remote Procedure call Flow

In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language. In both cases, you start with one or more parameters and you get back a result. This observation has led people to try to arrange request-reply interactions on networks to be cast in the form of procedure calls. Such an arrangement makes network applications much easier to program and more familiar to deal with. For example, just imagine a procedure named *get IP address (host name)* that works by sending a UDP packet to a DNS server and waiting for the reply, timing out and trying again if one is not forthcoming quickly enough. In this way, all the details of networking can be hidden from the programmer. The major work in this area was done by Birrell and Nelson

(1984), When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on machine 2. Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer. This technique is known as **RPC (Remote Procedure Call)** and has become the basis for many networking applications. Traditionally, the calling procedure is known as the client and the called procedure is known as the server, and we will use those names here too.

The idea behind RPC is to make a remote procedure call look as much as possible like a local one. In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub** that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local. The actual steps in making an RPC are shown in Figure.

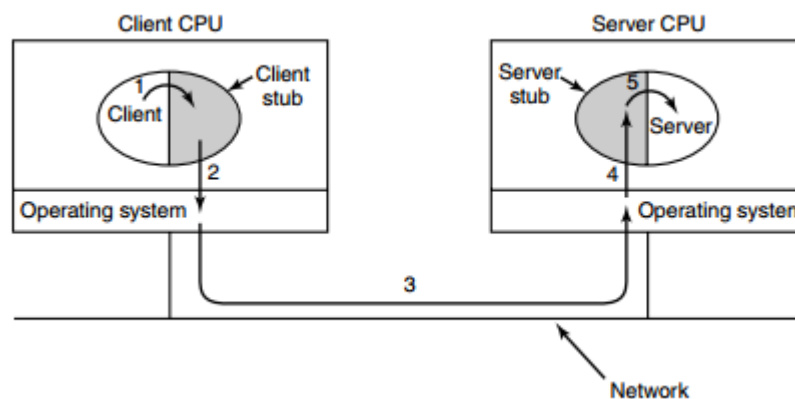


Figure: Steps in making a remote procedure call.

Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.

Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.

Step 3 is the operating system sending the message from the client machine to the server machine.

Step 4 is the operating system passing the incoming packet to the server stub.

Step 5 is the server stub calling the server procedure with the un-marshaled parameters. The reply traces the same path in the other direction.

The key item to note here is that the client procedure, written by the user, just makes a normal (i.e., local) procedure call to the client stub, which has the same name as the server

procedure. Since the client procedure and client stub are in the same address space, the parameters are passed in the usual way. Similarly, the server procedure is called by a procedure in its address space with the parameters it expects. To the server procedure, nothing is unusual. In this way, instead of I/O being done on sockets, network communication is done by faking a normal procedure call. Despite the conceptual elegance of RPC, there are a few snakes hiding under the grass. A big one is the use of pointer parameters. Normally, passing a pointer to a procedure is not a problem. The called procedure can use the pointer in the same way the caller can because both procedures live in the same virtual address

Remote Procedure Call (RPC) provides a different paradigm for accessing network services. Instead of accessing remote services by sending and receiving messages, a client invokes services by making a local procedure call. The local procedure hides the details of the network communication.

When making a remote procedure call

The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.

When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

The main goal of RPC is to hide the existence of the network from a program. As a result, RPC doesn't quite fit into the OSI model:

The message-passing nature of network communication is hidden from the user. The user doesn't first open a connection, read and write data, and then close the connection. Indeed, a client often doesn't even know they are using the network.

RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often. For example, on (diskless) Sun workstations, every file access is made via an RPC.

Advantages of RPC

1. RPC supports process oriented and thread oriented models
2. The internal message passing mechanism of RPC is hidden from User.
3. The Effort to re-write and re-develop the code is minimum in RPC.
4. RPC can be used in distributed environment as well as local environment.
5. Many of the protocol layers are omitted by RPC to improve the performance.
6. Supports multiple servers having same interface type.
7. Client requests can be spread evenly to balance the load.

Disadvantages of RPC

1. The overhead involved in binding clients to servers is large and becomes significant when many client processes are short lived.
2. A binding agent must be robust against failures and should not become performance bottleneck.

3. The RPC is a concept that can be implemented in different ways. It is not a standard.
4. There is no flexibility in RPC for hardware architecture, it is only interaction based.
5. There is increase in cost because of RPC.

RPC is especially well suited for client-server (e.g., query-response) interaction in which the flow of control alternates between the caller and callee. Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again. The following steps take place during an RPC:

A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client's own address space. The client stub marshals the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.

The client stub passes the message to the transport layer, which sends it to the remote server machine. On the server, the transport layer passes the message to a server stub, which de-marshals the parameters and calls the desired server routine using the regular procedure call mechanism. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshals the return values into a message. The server stub then hands the message to the transport layer.

The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub. The client stub de-marshals the return parameters and execution returns to the caller.

6.5 PRESENTATION LAYER

The presentation layer is 6th layer of OSI reference model. The presentation layer mainly translates data between the application layer and the network format. Data can be communicated in different formats via different sources. Thus, the presentation layer is responsible for integrating all formats into a standard format for efficient and effective communication.

The presentation layer follows data programming structure schemes developed for different languages and provides the real-time syntax required for communication between two objects such as layers, systems or networks. The data format should be acceptable by the next layers; otherwise, the presentation layer may not perform correctly.

6.5.1 Design issues: Presentation Layer

I. Standard way of encoding data

The presentation layer follows a standard way to encode data when it needs to be transmitted. This encoded data is represented as character strings, integers, floating point numbers, and data structures composed of simple components. It is handled differently by different machines based on the encoding methods followed by them.

II. Maintaining the Syntax and Semantics of distributed information

The presentation layer manages and maintains the syntax as well as logic and meaning of the information that is distributed.

III. Standard Encoding on the wire

The data structures that are defined to be exchanged need to be abstract along with the standard encoding to be used “on the wire”.

6.5.2 Functions: Presentation layer

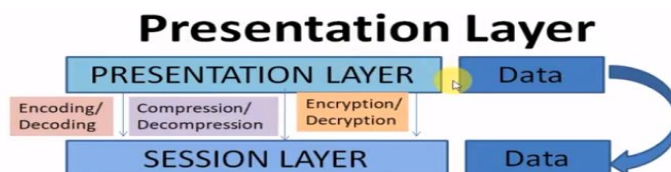
The functions of the presentation layer include the following:

1. Syntax conversion - The abstract syntax is converted to the transfer syntax, and the other side to achieve the opposite conversion. Involved in the contents of the code conversion, character conversion, data format modification, as well as data structure operation adaptation, data compression, encryption and so on.
2. Semantic negotiation - According to the requirements of the application layer to negotiate the appropriate choice of context, that is, to determine the transmission syntax and transmission.
3. Network security and confidentiality management, text compression and packaging, virtual terminal protocol (VTP).
4. Connection management - Including the use of the session layer service to establish a connection, manage data transport and synchronization control over this connection (using the corresponding services at the session level), and terminate the connection either normally or absently.

In reference to above the presentation layer used to perform 3 actions on the data i.e. transmitted from application layer to presentation layer

1. Encoding/decoding
2. Compression/Decompression
3. Encryption/decryption

In short, presentation layer decides that how the data will be transmitted to session while sending or to application while receiving.



6.5.2 Encoding/Decoding

Computers use encoding schemes to store and retrieve meaningful information as data. This is known as data encoding.

Data Encoding /Decoding is a process performed to give a data a standard format like what type of file it is, e.g we have to send a pdf over Bluetooth to another device then unless encoding is not done properly the pdf reader will not open the file at the receiver end. Hence, this is needed run the program. Another e.g. if we have to send a picture of jpeg format then the presentation layer will encode it so that later at the receiver's end presentation layer will decode it and the receiver is receiving the jpeg format file.

On electronic devices like computers, data encoding involves certain coding schemes that are simply a series of electrical patterns representing each piece of information to be stored and retrieved. For instance, a series of electrical patterns represents the letter "A." Data encoding and decoding occur through electronic signals, or the electric or electromagnetic encoding of data. In data encoding, all data is serialized, or converted into a string of ones and zeros, which is transmitted over a communication medium like a phone line. Serialization must be done in such a way that the computer receiving the data can convert the data back into its original format," according to Microsoft. "How serialization is accomplished is called a *communication protocol*, and is controlled by both software and data-transmission hardware. There are several levels at which the data is converted."

6.5.3 Compression/Decompression

Data compression reduces the size of a file by minimizing redundant data. In a text file, redundant data can be frequently occurring characters, such as the space character, or common vowels, such as the letters e and a; it can also be frequently occurring character strings. Data compression creates a compressed version of a file by minimizing this redundant data.

Each type of data-compression algorithm minimizes redundant data in a unique manner. For example, the **Huffman encoding algorithm** assigns a code to characters in a file based on how frequently those characters occur. Another algorithm, called **RPL (run-length encoding)**, generates a two-part value for repeated characters: the first part specifies the number of times the character is repeated, and the second part identifies the character.

6.5.4 Encryption/Decryption

This process is used for data protection while sending and receiving the data, used in high security transfers. Like passwords are used in files to open is also an encryption technique but in network encryption coding a data with specific code so that if someone goes through the data i.e Hack no one may not be able to access it **because of encryption**.

The receiver will receive exactly the same data that was transmitted by sender **because of decryption**.

The presentation layer is concerned with preserving the meaning of information sent across a network. The presentation layer may represent (encode) the data in various ways (e.g.,

data compression, or encryption), but the receiving peer will convert the encoding back into its original meaning. The syntax and the semantics of the information exchanged between two communication systems are managed by the presentation layer of the OSI Model.

6.6 DATA COMPRESSION

Data compression is a reduction in the number of bits needed to represent data. Compressing data can save storage capacity, speed file transfer, and decrease costs for storage hardware and network bandwidth. Data compression refers to the reducing the number of bits that need to be transmitted over communication channel.

Data compression reduces the number of bits sent. Data compression becomes particularly important when we send data with high size such as audio & video. Even with very fast transmission speed of data we need to send data in short time. Virtually all form of data contains redundancy i.e. it is the amount of wasted “space” used to transmit certain data. By making use of more efficient data representation methods, redundancy can be reduced. Even 7 bit ASCII code has some redundancy in it. The goal of data compression is to represent an information source (e.g. a data file, a speech signal, an image, or a video signal) as accurately as possible using the fewest number of bits.

Compression Ratio

Data compression ratio, also know as compression power, is a term used to quantify the reduction in data-representation size produced by a data compression algorithm. The data compression ratio is analogous to the physical compression ratio used to measure physical compression of substances.

Data compression ratio is defined as the ratio between the uncompressed size and compressed size.

$$CompressionRatio = \frac{UncompressedSize}{CompressedSize}$$

Thus a representation that compresses a 20 MB file to 2 MB has a compression ratio of $20/2 = 10$, often noted as an exploit ratio, 10:1 (read “ten” to “one”) or as an implicit ratio 10/1.

Note that this formulation applies equally for compression, where the uncompressed size is that of the original and for decompression, where the uncompressed size is that of the reproduction.

Sometimes the space saving is given instead, which is defined as the reduction in size relative to the uncompressed size.

$$SpaceSaving = 1 - \frac{CompressedSize}{UncompressedSize}$$

Thus a representation that compresses a 20 MB file to 2 MB would yield a space saving of $1 - 2/20 = 0.9$ often notated as a percentage, 90%.

For signals of indefinite size, such as streaming audio and video the compression ratio is defined in terms of uncompressed and compressed data rates instead of data sizes.

$$\text{CompressionRatio} = \frac{\text{Uncompressed DataRate}}{\text{CompressedDataRate}}$$

Also, Data-rate saving, this is computed in reference to the compressed data rate and uncompressed data rate.

$$\text{DataRateSaving} = 1 - \frac{\text{CompressedDataRate}}{\text{UncompressedDataRate}}$$

6.6.1 Data Compression Methods

The data compression methods are of two types:

A. Lossy Compression: removes non-useful part of data that is undetectable. It decreases the size of file to a greater extent and in process quality of data is degraded.

B. Lossless Compression: reconstructs the exact data. It decreases size but at low extent and doesn't degrade the quality of data.

Lossy Compression

Lossy compression refers to data compression techniques in which some amount of data is lost. Lossy compression technologies attempt to eliminate redundant or unnecessary information. If the decompressed information need not be an exact replica of the original information but something very close, we can use a lossy data compression method.

Lossy compression reduces a file by permanently eliminating certain information, especially redundant information. When the file is uncompressed, only a part of the original information is still there(although the user may not notice it). Lossy compression is generally used for video and sound, where a certain amount of information loss will not be detected by most users.

Different Lossy Techniques: Methods Based on Lossy compression.

JPEG: Used for pictures and Graphics

MPEG: used for Video Compression.

Audio Compression

Comparatively, these methods are less time taking, cheaper as well as it can reduce more space.

The JPEG image file, commonly used for photographs and other complex still images on the web, is an image that has lossy compression. Using JPEG compression, the creator can decide how much loss to introduce and make a trade-off between file size and image quality. Most video compression technologies, such as MPEG, use Lossy technique.

The best example is a videoconference where there is an acceptable amount of frame loss in order to deliver the image in real time. People may appear jerky in their movements, but you still have a grasp for what is happening on the other end of the conference. In the case of

graphics files, some resolution may be lost in order to create a smaller file. The loss may be in the form of color depth or graphic detail. For example, high-resolution details can be lost if a picture is going to be displayed on a low-resolution device. Loss is also acceptable in voice and audio compression, depending on the desired quality

These methods are called lossy compression methods because we will lose some of the original data in the process. Several methods have been developed using lossy compression techniques. Joint Photographic experts group(JPEG) is used to compress graphics and pictures. Motion Picture Expert Group (MPEG) is used to compress video.

Lossless Compression

The lossless compression refers to data compression techniques in which no data is lost. For most types of data, lossless compression techniques can reduce the space needed by only about 50%.

For greater compression, one must use a lossy compression technique. Note, however, that only certain types of data-graphics, audio, and video can tolerate lossy compression.

We must use a lossless compression technique when compressing data and programs. The PKZIP compression technology is an example of lossless compression. With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. All the information is completely restored.

This is generally the technique of choice for text or spreadsheet files, where losing words or financial data could pose a problem. The Graphics Interchange File(GIF) is an image format used on the Web that provides lossless compression. In lossless Data Compression the compressing & decompressing algorithm are usually the inverse of each other. In other words after decompressing we will get exact data as they were before compressing, Nothing is lost. The following are some techniques used in lossless data compression.

1. Null compression
2. Run Length Encoding
3. Statistical Compression
 - Shannon-Fano Encoding
 - Huffman Encoding
 - Adaptive compression(Adaptive Huffman)
 - Dictionary Based Compression
 - ❖ LZ77, LZ8 Algorithms
 - ❖ Lempel Ziv Welch(LZW) encoding

Null Compression

It replaces a series of blank with a compression code, followed by a value that represents the number of spaces.

Example: hello friend how is your life?

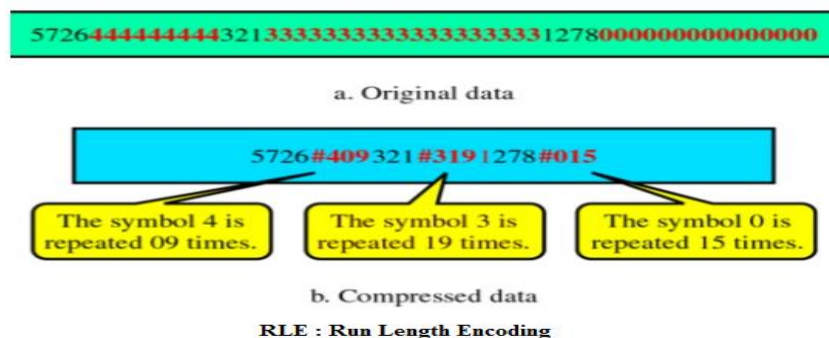
When data contain strings of repeated symbols (such as bits or characters), the strings can be replaced by special marker, followed by the repeated symbol, followed by the number of occurrences. For example, in given Figure, the symbol # is the marker. The symbol being repeated (the run symbol) follows the marker.

After the run symbol, the number of occurrences (length) is shown by a two digit number.
 Example: This run-length encoding mentioned can be used in audio (silence is a run of 0s) and video (run of picture element having the same brightness and color).

6.7 RUN LENGTH ENCODING

Simple Compression technique Replace all consecutive numbers or alphabets by first the number of times an alphabet was used followed by the alphabet itself. This method becomes more effective with numbers especially when it is about only bits 1 and 0

When data contain strings of repeated symbols (such as bits or characters), the strings can be replaced by a special marker, followed by the repeated symbol, followed by the number of occurrences. For example, in given figure, the symbol # is the marker. The symbol being repeated (the run symbol) follows the marker. After the run symbol, the number of occurrences (length) is shown by a two-digit number.



Statistical Compression

Another method of lossless compression is statistical compression. This method uses short codes for frequent symbols and long codes for infrequent symbols. In this way, the length of the total data is reduced tremendously. The following methods are popular lossless statistical compression methods:

a. Shannon Fano Algorithm

Shannon Fano Algorithm is an entropy encoding technique for lossless data compression of multimedia. Named after Claude Shannon and Robert Fano, it assigns a code to each symbol based on their probabilities of occurrence. It is a variable length encoding scheme, that is, the codes assigned to the symbols will be of varying length.

HOW DOES IT WORK?

The steps of the algorithm are as follows:

1. Create a list of probabilities or frequency counts for the given set of symbols so that the relative frequency of occurrence of each symbol is known.
2. Sort the list of symbols in decreasing order of probability, the most probable ones to the left and least probable to the right.
3. Split the list into two parts, with the total probability of both the parts being as close to each other as possible.
4. Assign the value 0 to the left part and 1 to the right part.

5. Repeat the steps 3 and 4 for each part, until all the symbols are split into individual subgroups.

The Shannon codes are considered accurate if the code of each symbol is unique.

b. Huffman Coding

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters; lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

The variable-length codes assigned to input characters are prefix code means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit-stream.

Let us understand prefix codes with a counter example. Let there be four characters a, b, c and d, and their corresponding variable length codes be 00, 01, 0 and 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be “cccd” or “ccb” or “acd” or “ab”.

There are mainly two major parts in Huffman Coding

1. Build a Huffman Tree from input characters.
2. Traverse the Huffman Tree and assign codes to characters.

Steps to build Huffman Tree

Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)
2. Extract two nodes with the minimum frequency from the min heap.
3. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.
4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

Let us understand the algorithm with an example:

for example we have a sentence like :- “ Data Communication Application”

We have to assign numbers to each alphabet and count occurrences:-

D	A	T	C	O	M	U	N	I	P	L
1	5	3	3	3	2	1	3	4	2	1

c. Dictionary compression

The compression techniques we have seen so far replace individual symbols with variable length code-words. In dictionary compression, variable length substrings are replaced by short, possibly even fixed length code-words. Compression is achieved by replacing long strings with shorter code-words. The general scheme is as follows:

- The dictionary D is a collection of strings, often called phrases. For completeness, the dictionary includes all single symbols.

- The text T is parsed into a sequence of phrases:

$$T = T_1T_2 \dots T_z, T_i \in D.$$

The sequence is called a parsing or a factorization of T with respect to D .

- The text is encoded by replacing each phrase T_i with a code that acts as a pointer to the dictionary.

Here is a simple static dictionary encoding for English text:

- The dictionary consists of some set of English words plus individual symbols.
- Compute the frequencies of the words in some corpus of English texts. Compute the frequencies of symbols in the corpus from which the dictionary words have been removed.
- Number the words and symbols in descending order of their frequencies.
- To encode a text, replace each dictionary word and each symbol that does not belong to a word with its corresponding number. Encode the sequence of number using γ coding.

Lempel-Ziv compression In 1977 and 1978, Abraham Lempel and Jacob Ziv published two adaptive dictionary compression algorithms are known as LZ77 and LZ78, and most related methods can be categorized as a variant of one or the other. The primary difference is the encoding of the phrases:

- LZ77 uses direct pointers to the preceding text.
- LZ78 uses pointers to a separate dictionary.

6.8 CRYPTOGRAPHY

The word Cryptography has been taken from greek words “kryptos” which means hidden and graphein which means to write. It simply means the design and analysis of codes and ciphers.

Cryptography is the process of transforming plain text or original information into an unintelligible form (cipher text) so that it may be sent over unsafe channels of communication.

6.8.1 Public Key Cryptography

Public Key Cryptography Both parties have a private key and a public key. The private keys are known only to their owners, but the public keys are available to anyone (like telephone numbers). The sending party encrypts the message with the receivers public key and the receiver decrypts with his own private key. This is possible due to the discovery by Diffie and Hellman (at Stanford University, autumn 1975) that algorithms can be developed which use one key for encryption and a different key for decryption. The public and private key constitute a key pair. The following public key crypto-systems are well known:

- The RSA (named after it's inventors Rivest, Shamir and Adleman) algorithm was

developed at MIT in 1977 and is the most common public key system in use today. A key minimum key length of 768 bits is recommended by RSA Inc. RSAREF is a library from RSA Inc. which is integrated into many commercial products and public domain products (such as the US version of PGP). International, public domain RSA compatible libraries (with large key sizes) do exist and are used in products such as SSH (SSH uses 1024 bit keys by default). RSA key generation is slower than verification. RSA is patented in the U.S. until 20.9.2000.

- The Diffie-Hellman (named also after its inventors) key exchange protocol, published in 1976, produces shared secret keys from publicly known information over unsecured networks. These shared keys can be used to produce session keys. Its strength is based on the "discrete logarithm" problem. Since parties are not authenticated, it is vulnerable to "man in the middle" attacks, which can be prevented by use of additional protocols or digital signatures. Sun makes extensive use of this algorithm in Secure RPC and SKIP. This algorithm has the added advantage that its patent expired in 1997.

- The ElGamal Public Key system (invented by Taher ElGamal) consists of both an encryption and signature algorithm. It is similar to the Diffie-Hellman key exchange and its strength is based on the "discrete logarithm" problem. Key length strengths are similar to RSA. It is quite slow and requires very good random number generation. DSA is based on the signature algorithm.

DSS is the Digital Signature Standard that uses the DSA (Digital Signature Algorithm) approved in May 1994 by the U.S. government (NIST & NSA) as the standard for digital authentication. DSA is based on crypto algorithms from ElGamal and Schnorr. Signature generation is faster than verification (which is unusual, there are likely to be more verifications than generations). DSA lacks a key exchange mechanism, is very new and has been criticized because the NSA were heavily involved in its selection and it was not subject to open peer review.

6.8.2 Cryptography Features

1. Confidentiality:

Information can only be accessed by the person for whom it is intended and no other person except him can access it.

2. Integrity:

Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

3. Non-repudiation:

The creator/sender of information cannot deny his or her intention to send information at a later stage.

4. Authentication:

The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

6.8.3 Types of Cryptography

In general there are three types of cryptography:

Symmetric Key Cryptography, Hash Functions and Asymmetric Key Cryptography

1. Symmetric Key Cryptography

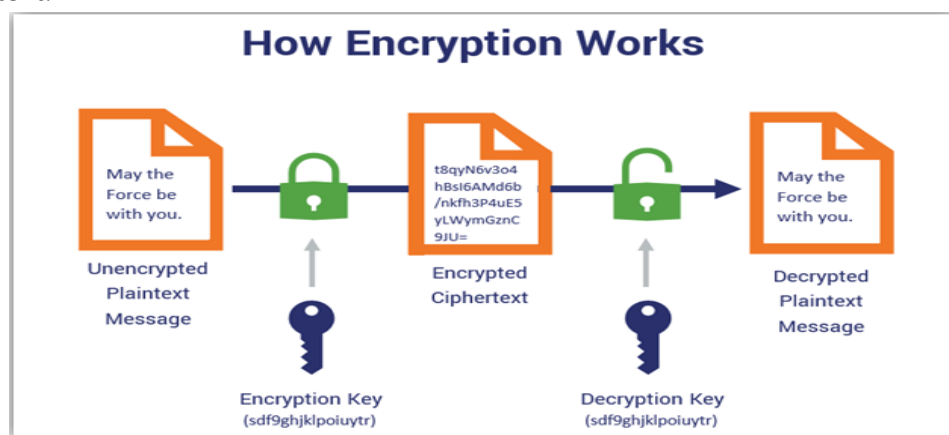
It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System(DES).

A system where one secret key shared is called symmetric or secret key Cryptography. An encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. Contrast this with public-key cryptology, which utilizes two keys - a public key to encrypt messages and a private key to decrypt them. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. Public-key encryption avoids this problem because the public key can be distributed in a non-secure way and the private key is never transmitted. Symmetric-key cryptography is sometimes called secret-key cryptography. The most popular symmetric-key system is the Data Encryption Standard (DES).

To put this in the simplest terms possible, **symmetric encryption** is a type of encryption that uses the same key to encrypt and decrypt data. Both the sender and the recipient have identical copies of the key, which they keep secret and don't share with anyone. This differs from **asymmetric encryption**, which uses two keys — a public key (that anyone can access) to encrypt information and a private key to decrypt information.

Just in case it's helpful, let's do a quick review of how encryption works in general:

1. The sender uses an encryption key (usually a string of letters and numbers) to encrypt their message.
2. The encrypted message, called cipher-text, looks like scrambled letters and can't be read by anyone along the way.
3. The recipient uses a decryption key to transform the cipher-text back into readable text.



Only these two parties (sender and recipient) can read and access the data. This is why it's also sometimes called **secret key encryption**, **secret key cryptography**, **private key cryptography**, **symmetric cryptography** and **symmetric key encryption**.



Having only one key to serve both the encryption and decryption functions simplifies the encryption process. After all, you're applying one key to turn plaintext, readable information into unreadable gibberish (ciphertext) and vice versa. One of the advantages of using symmetric encryption is that it provides data privacy and confidentiality without the extra complexity of multiple keys.

Symmetric key encryption does work on its own, for certain use cases. For example, it's useful for encrypting databases and files, where you're not exchanging data publicly between parties. But as with any technical process, there are other advantages and disadvantages of using symmetric key encryption, such as key distribution and management issues, and we'll talk about those a little later.

2. Hash Functions

There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

A "Hash function" is a complex encryption algorithm used primarily in cryptography and is like a shortened version of full-scale encryption. A hash function H is a transformation that takes a variable-size input m and returns a fixed-size string, which is called the hash value h (that is, $h = H(m)$). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography the hash functions are usually chosen to have some additional properties.

The basic requirements for a cryptographic hash function are:

- the input can be of any length,
- the output has a fixed length,
- $H(x)$ is relatively easy to compute for any given x ,
- $H(x)$ is one-way,
- $H(x)$ is collision-free.

A hash function H is said to be one-way if it is hard to invert, where "hard to invert" means that given a hash value h , it is computationally infeasible to find some input x such that $H(x) = h$.

If, given a message x , it is computationally infeasible to find a message y not equal to x such that $H(x) == H(y)$ then H is said to be a weakly collision-free hash function.

A strongly collision-free hash function H is one for which it is computationally infeasible to find any two messages x and y such that $H(x) = H(y)$.

The hash value represents concisely the longer message or document from which it was computed; one can think of a message digest as a digital fingerprint of the larger document. Examples of well-known hash functions are MD2 and MD5 and SHA.

Perhaps the main role of a cryptographic hash function is in the provision of digital signatures. Since hash functions are generally faster than digital signature algorithms, it is typical to compute the digital signature to some document by computing the signature on the document's hash value, which is small compared to the document itself. Additionally, a digest can be made public without revealing the contents of the document from which it is derived. This is important in digital time stamping where, using hash functions, one can get a document time stamped without revealing its contents to the time stamping service.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs) and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums or just hash values, even though all these terms stand for functions with rather different properties and purposes.

3. Asymmetric Key Cryptography

Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

In asymmetric cryptography (also known as public key cryptography), a public key and private key are used as a two-key system which allows for free distribution of the encryption key while making the decryption key public. This gives the benefit of allowing-a person to give away the encryption key to anyone while still keeping the data enciphered and sent to them secure. Even if someone intercepts the message and has a copy of the public key, they can't access the data. This is the major benefit of using the two-key system rather than one-key. A graphical explanation of this is shown to the right. Asymmetric key algorithms can also be used in the reverse- proving someone's identity to a large group of people. Suppose a big political leader wants to make announcements via the internet, but wants to ensure against imposters. He would give away the public key to everyone, encrypt data with the private key and send out the cipher-text via the internet. Since he has the only private key, only messages that were actually from him would be readable with his corresponding public key.

Mathematically, asymmetric encryption is considerably more complicated than single-key ciphers. The basic idea is to make computations which are incredibly hard without the key, but simple when you have the key available. With RSA (and several other 2-key ciphers) this is done with a pair of very large prime numbers. With the public key and these two large primes, the calculation of the private key (and therefore the decryption of the ciphertext) is simple. However, without the two primes, the calculation is incredibly complicated, with

millions or billions of possibilities. This renders the public key useless in decryption, but accessible to encryption functionality on its own. This use of two key encryption is the basis for digital signatures and certificates as well. Digital signatures are essentially an assurance of authentication and nonrepudiation by tying a public key to a private key. Just like its physical counterpart (theoretically), a digital signature is an undeniable proof of identity- it can't be forged and therefore cannot be denied at a later time by its owner. Digital signatures would be done by using both sets (sender and receiver) of public private keys. The sender would encrypt a message using the receiver's public key so only the receiver can read it. He would then encrypt the ciphertext "message" with his private key, so anyone who decrypts the message knows it comes from him. The message is sent and decrypted twice, once for authentication/nonrepudiation and again for confidentiality. Certificates are similar, but use a certificate authority (CA) which acts as a trusted third party or voucher, to verify identity. Certificates have more details, however, giving the user's name, the CA, expiration date (if applicable) and any approved operations. This is a major aspect in modern e-commerce systems, where companies such as Verisign act as a reputable 3rd party, issuing certificates to companies.

Types of Asymmetric Cryptology

RSA asymmetric encryption

RSA is the best known asymmetric (public key) algorithm, named after its inventors: Rivest, Shamir and Adleman. RSA uses public and private keys that are functions of a pair of large prime numbers. Its security is based on the difficulty of factoring large integers. The RSA algorithm can be used for both public key encryption and digital signatures. The keys used for encryption and decryption in RSA algorithm, are generated using random data. The key used for encryption is a public key and the key used for decryption is a private key. Public keys are stored anywhere publicly accessible. The sender of message encrypts the data using public key and the receiver decrypts it using his/her own private key. That way, no one else can intercept the data except receiver.

Digital Signature Algorithm -DSA

The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS).

PGP

PGP (Pretty Good Privacy) is a public-private key cryptography system which allows for users to more easily integrate the use of encryption in their daily tasks, such as electronic mail protection and authentication and protecting files stored on a computer. PGP was originally designed by Phil Zimmerman. It uses IDEA, CAST or Triple DES for actual data encryption and RSA (with up to 2048-bit key) or DH/DSS (with 1024-bit signature key and 4096-bit encryption key) for key management and digital signatures. The RSA or DH public key is used to encrypt the IDEA secret key as part of the message.

6.8.4 Cryptology terminologies

1) Crypto system - It is a system of a secure key pair consisting of a private key, for creating a digital signature and a public key to verify the digital signature.

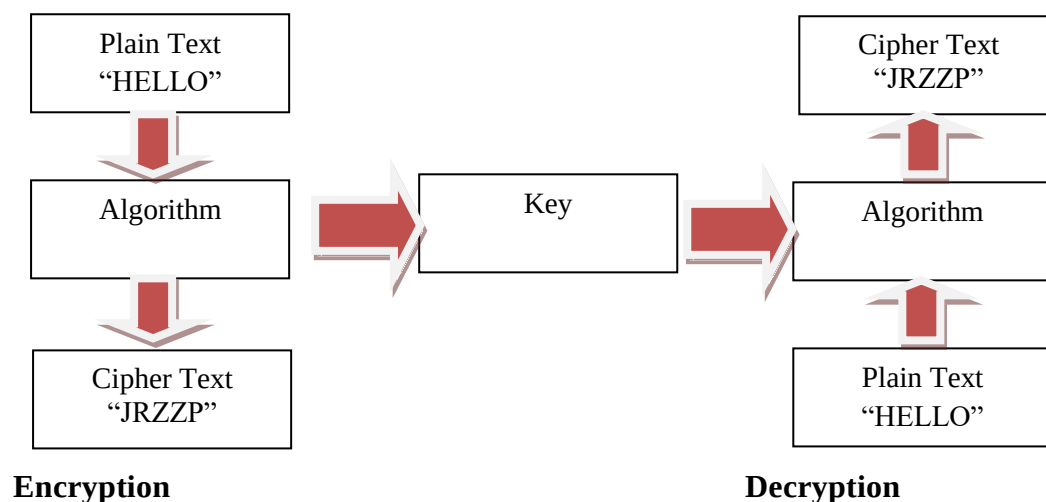
2) Key pair - It means a private key and its mathematically related public key, which are so related that the public key can verify a digital signature created by the private key.

3) Private key - It means the key of a key pair used to create a digital signature.

4) Public key - It means the key of a key pair used to verify a digital signature and listed in the Digital Signature certificate.

5) Hash function - It means an algorithm mapping or translation of one sequence of bits into another, generally smaller, set known as "hash result" such that an electronic record yields the same hash result every time the algorithm is executed with the same electronic record as its input making it computationally infeasible (a) to derive or reconstruct the original electronic record from the hash result produced by the algorithm. (b) That two electronic records can produce the same hash result using the algorithm.

The second stage for cryptology is decryption (crypto analysis). Here, the cipher text back into the original form, when done by the authorized person.



Modern cryptographic systems are implemented with computer programs that have two inputs- the plaintext message and key, both of which are represented as sequences of 0s and 1s.

6.9 REVIEW QUESTIONS

1. What are the challenges of RPC?

2. What is the significance of client stub and server stub?
3. Differentiate between Compression and Decompression.
4. How compression ratio for finite and indefinite data sizes is different?
5. Design a Huffman coding for “**JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY**”.
6. What is the difference between network security and cryptography?

6.10 SUMMARY

In this unit, we have given you a overview of various features and issues of session and presentation layers of OSI reference model. Our discussions are focused on various services like encoding, compression and encryption techniques with examples for better understanding.

REFERENCES FOR FURTHER READINGS

1. BehrouzForouzan, Data communications and Networking, Tata Mc-Graw Hill.
2. Andrew S. Tanenbaum, Computer Networks, Pearson Education.
3. William Stallings, Data and Computer Communications, Pearson education.

UNIT 7: APPLICATION LAYER

STRUCTURE

- 7.0 Introduction**
- 7.1 Objectives**
- 7.2 Concepts**
- 7.3 Application Layer Functions**
- 7.4 Application Layers Services**
- 7.5 Application Layer Protocols**
 - 7.5.1 Domain Name System (DNS)**
 - 7.5.2 HTTP**
 - 7.5.3 Uniform Resource Locator (URL)**
 - 7.5.4 E-mail**
 - 7.5.5 WWW**
 - 7.5.6 Telnet**
 - 7.5.7 File Transfer Protocol (FTP)**
 - 7.5.8 SMTP**
- 7. 6 Review Questions**
- 7.7 Summary**
- 7.8 References for Further Readings**

7.0 OBJECTIVES

At the end of this unit, you will be able to:

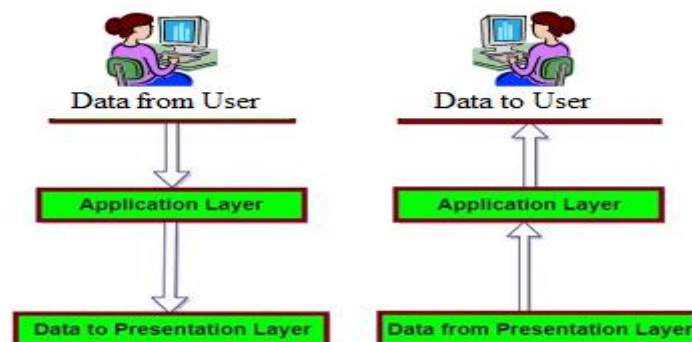
- Understand the significance of application layer in data transmission.
- Understand the functions & services of application layer.
- Discuss in detail the various application layer protocols.

7.1 INTRODUCTION

Application layer is the top most layer in OSI and TCP/IP layered model. This layer exists in both layered Models because of its significance of interacting with user and user applications. This layer provides the better interface that helps to directly interacts along with software application and offers common web application services. This layer also fires a request to presentation layer, and it also delivers the network services to end-users.

7.2 CONCEPTS

A user may or may not directly interact with the applications. Application layer is where the actual communication is initiated and reflects. Because this layer is on the top of the OSI layer stack, it does not serve any other layers. Application layer takes the help of Transport and all layers below it to communicate or transfer its data to the remote host.



When an application layer protocol wants to communicate with its peer application layer protocol on remote host, it hands over the data or information to the transport layer. The transport layer does the rest with the help of all the layers below it.

Two remote application processes can communicate mainly in two different ways:

- **Peer-to-peer:** Both remote processes are executing at same level and they exchange data using some shared resource.
- **Client-Server:** One remote process acts as a Client and requests some resource from another application process acting as Server.

In client-server model, any process can act as Server or Client. It is not the type of machine, size of the machine, or its computing power which makes it server; it is the ability of serving request that makes a machine a server.

A system can act as Server and Client simultaneously. That is, one process is acting as Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

7.3 APPLICATION LAYER FUNCTIONS

Application layer allows to users to implement the all services of network.

- This layer helps to design of network-based applications.
- It allows to users several services such as user login, transmission of data, e-mail, network devices configuration, and messages formatting.
- It is enabled with error handling and recovery of information.
- This layer offers various rights like as retrieve and handle files from remote terminal system.
- It provides the virtual terminal that take care abstract data structure, and it is controlled by keyboard and your computer system which helps to display your current state of data structure.
- It provides the authentication in between multiple network devices for delivering most security in the layer.
- Application layer also offers the mail services like as email forwarding and storage of e-mails.
- It also provides the directory services that help to access global information related to several objects and services.
- This layer monitors the sufficient availability of resources which are need in the network.

7.4 APPLICATION LAYERS SERVICES

- **Network Virtual terminal:** An application layer allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user's computer talks to the software terminal, which in turn, talks to the host. The remote host thinks that it is communicating with one of its own terminals, so it allows the user to log on.
- **File Transfer, Access, and Management (FTAM):** An application allows a user to access files in a remote computer, to retrieve files from a computer and to manage files in a remote computer. FTAM defines a hierarchical virtual file in terms of file structure, file attributes and the kind of operations performed on the files and their attributes.
- **Addressing:** To obtain communication between client and server, there is a need for addressing. When a client made a request to the server, the request contains the server address and its own address. The server response to the client request, the request contains the destination address, i.e., client address. To achieve this kind of addressing, DNS is used.
- **Mail Services:** An application layer provides Email forwarding and storage.
- **Directory Services:** An application contains a distributed database that provides access for global information about various objects and services.

- **Authentication:** It authenticates the sender or receiver's message or both.

7.5 APPLICATION LAYER PROTOCOLS

An application layer protocol defines how application processes (clients and servers), running on different end systems, pass messages to each other. In particular, an application layer protocol defines:

- The types of messages, e.g., request messages and response messages.
- The syntax of the various message types, i.e., the fields in the message and how the fields are delineated.
- The semantics of the fields, i.e., the meaning of the information that the field is supposed to contain;
- Rules for determining when and how a process sends messages and responds to messages.

Application layer protocols can be broadly divided into two categories:

- 1) Protocols which are used by users. For example, email.
- 2) Protocols which help and support protocols used by users. For example DNS.

Some of Application layer protocols are DNS, HTTP, URL etc.

7.5.1 Domain Name System (DNS)

DNS (Domain Name System) is a decentralized naming system that converts domain names (such as logn.in) to its corresponding IP addresses. DNS is required for the functioning of the internet.

DNS is a service that translates the domain name into IP addresses. Each device connected to the internet has a unique IP address, which is a 32-bit number for example 132.32.0.1(in IPv4), and remembering such an IP address is almost impossible. The solution to this problem was using English letters, as we humans' beings are good at memorizing English words. Hence the concept of Domain Name came into the picture. Each domain name is a combination of English alphabets generally) and points to an IP address.

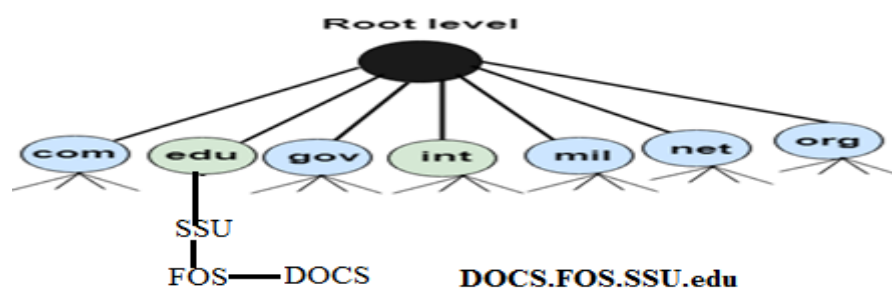
Label	Description
aero	Airlines and aerospace companies
biz	Businesses or firms
com	Commercial Organizations
coop	Cooperative business Organizations
edu	Educational institutions
gov	Government institutions

info	Information service providers
int	International Organizations
mil	Military groups
museum	Museum & other nonprofit organizations
name	Personal names
net	Network Support centers
org	Non-profit Organizations
pro	Professional individual Organizations

DNS is a TCP/IP protocol used on different platforms. The domain name space is divided into three different sections: generic domains, country domains, and inverse domain.

Generic Domains

It defines the registered hosts, according to their generic behaviour. Each node in a tree defines the domain name, which is an index to the DNS database. It uses three-character labels, and these labels describe the organization type.



Country Domain

The format of country domain is same as a generic domain, but it uses two-character country abbreviations (e.g., us for the United States) in place of three-character organizational abbreviations.

Inverse Domain

The inverse domain is used for mapping an address to a name. When the server has received a request from the client, and the server contains the files of only authorized clients. To determine whether the client is on the authorized list or not, it sends a query to the DNS server and ask for mapping an address to the name.

Working of DNS

- DNS is a client/server network communication protocol. DNS clients send requests to the. Server while DNS servers send responses to the client.

- Client requests contain a name which is converted into an IP address known as a forward DNS lookups while requests containing an IP address which is converted into a name known as reverse DNS lookups.
- DNS implements a distributed database to store the name of all the hosts available on the internet.
- If a client like a web browser sends a request containing a hostname, then a piece of software such as **DNS resolver** sends a request to the DNS server to obtain the IP address of a hostname. If DNS server does not contain the IP address associated with a hostname, then it forwards the request to another DNS server. If IP address has arrived at the resolver, which in turn completes the request over the internet protocol.

7.5.2 HTTP

HTTP stands for Hyper Text Transfer Protocol. It is a protocol used to access the data on the World Wide Web (www). It is the foundation of World Wide Web. The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.

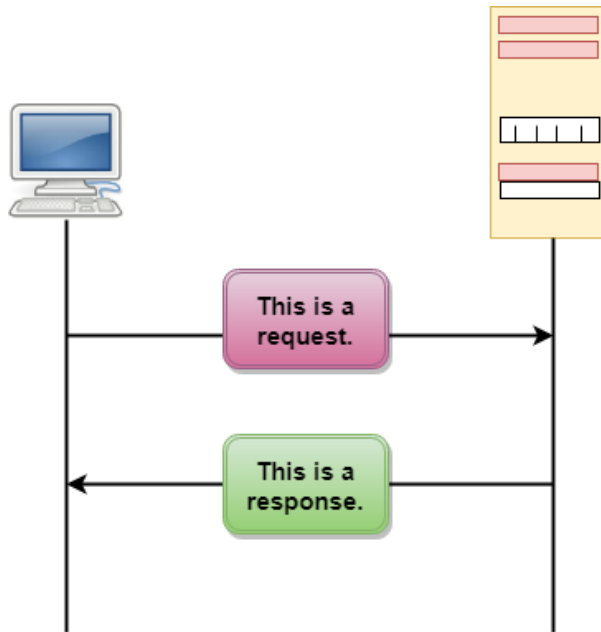
This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document. HTTP is similar to the FTP as it also transfers the files from one host to another host. But, HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files. HTTP is used to carry the data in the form of MIME-like format. HTTP is similar to SMTP as the data is transferred between client and server. The HTTP differs from the SMTP in the way the messages are sent from the client to the server and from server to the client. SMTP messages are stored and forwarded while HTTP messages are delivered immediately.

Features of HTTP:

- **Connectionless protocol:** HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- **Media independent:** HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- **Stateless:** HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

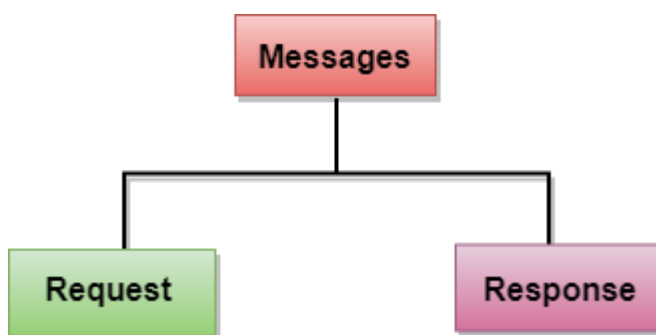
HTTP Transactions

The figure shows the HTTP transaction between client and server. The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

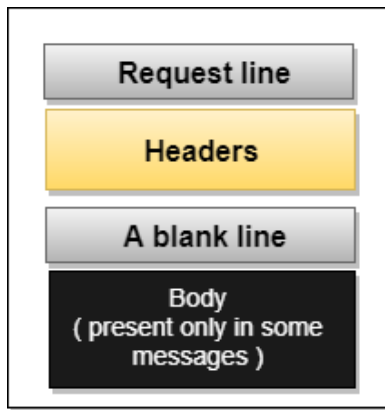


Messages

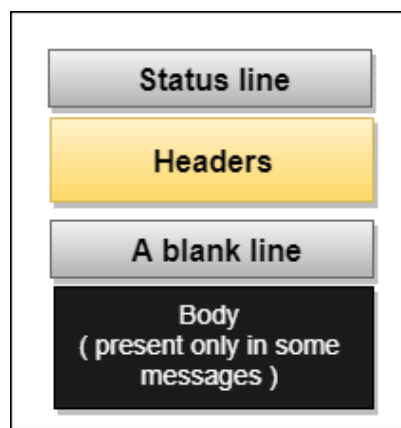
HTTP messages are of two types: request and response. Both the message types follow the same message format.



Request Message: The request message is sent by the client that consists of a request line, headers, and sometimes a body.



Response Message: The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



7.5.3 Uniform Resource Locator (URL)

A client that wants to access the document in an internet needs an address and to facilitate the access of documents, the HTTP uses the concept of Uniform Resource Locator (URL). The Uniform Resource Locator (URL) is a standard way of specifying any kind of information on the internet.

The URL defines four parts: method, host computer, port, and path.



- **Method:** The method is the protocol used to retrieve the document from a server. For example, HTTP.

- **Host:** The host is the computer where the information is stored, and the computer is given an alias name. Web pages are mainly stored in the computers and the computers are given an alias name that begins with the characters "www". This field is not mandatory.
- **Port:** The URL can also contain the port number of the server, but it's an optional field. If the port number is included, then it must come between the host and path and it should be separated from the host by a colon.
- **Path:** Path is the pathname of the file where the information is stored. The paths itself contain slashes that separate the directories from the subdirectories and files.

HTTP versions

- HTTP 1.0 uses non persistent HTTP. At most one object can be sent over a single TCP connection.
- HTTP 1.1 uses persistent HTTP. In this version, multiple objects can be sent over a single TCP connection.

7.5.4 Email

Email is a service which allows us to send the message in electronic mode over the internet. It offers an efficient, inexpensive and real time mean of distributing information among people.

E-Mail Address

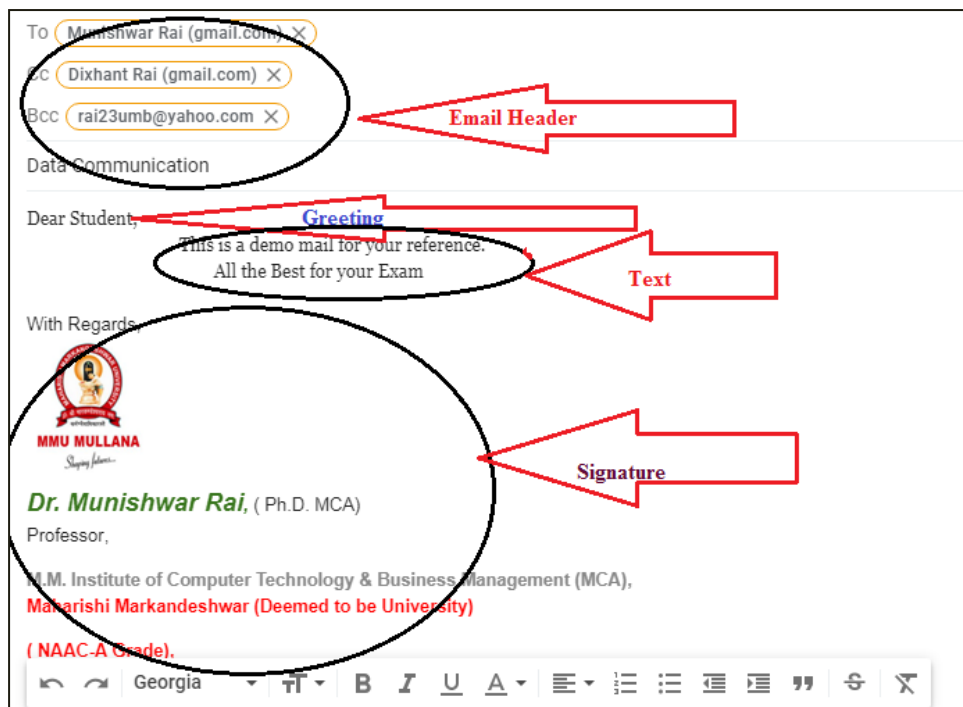
Each user of email is assigned a unique name for his email account. This name is known as E-mail address. Different users can send and receive messages according to the e-mail address.

E-mail is generally of the form `username@domainname`. For example, `connect@psou.ac.in` is an e-mail address where **connect** is username and **psou.ac.in** is domain name.

- The username and the domain name are separated by @ (**at**) symbol.
- E-mail addresses are not case sensitive.
- Spaces are not allowed in e-mail address.

E-mail Message Components

E-mail message comprises of different components: E-mail Header, Greeting, Text, and Signature. These components are described in the following diagram:



E-mail Header

The first five lines of an E-mail message is called E-mail header. The header part comprises of following fields:

From

The **From** field indicates the sender's address i.e. who sent the e-mail.

Date

The **Date** field indicates the date when the e-mail was sent.

To

The **To** field indicates the recipient's address i.e. to whom the e-mail is sent.

Subject

The **Subject** field indicates the purpose of e-mail. It should be precise and to the point.

CC

CC stands for Carbon copy. It includes those recipient addresses whom we want to keep informed but not exactly the intended recipient.

BCC

BCC stands for Blind Carbon Copy. It is used when we do not want one or more of the recipients to know that someone else was copied on the message.

Greeting

Greeting is the opening of the actual message. Eg. Hi Sir or Hi Guys etc.

Text

It represents the actual content of the message.

Signature

This is the final part of an e-mail message. It includes Name of Sender, Address, and Contact Number.

Advantages of Email

- ✓ **Reliable**
Many of the mail systems notify the sender if e-mail message was undeliverable.
- ✓ **Convenience**
There is no requirement of stationary and stamps. One does not have to go to post office. But all these things are not required for sending or receiving an mail.
- ✓ **Speed**
E-mail is very fast. However, the speed also depends upon the underlying network.
- ✓ **Inexpensive**
The cost of sending e-mail is very low.
- ✓ **Printable**
It is easy to obtain a hardcopy of an e-mail. Also an electronic copy of an e-mail can also be saved for records.
- ✓ **Global**
E-mail can be sent and received by a person sitting across the globe.
- ✓ **Generality**

It is also possible to send graphics, programs and sounds with an e-mail.

Disadvantages of Email

Apart from several benefits of E-mail, there also exist some disadvantages as discussed below:

- **Forgery**
E-mail doesn't prevent from forgery, that is, someone impersonating the sender, since sender is usually not authenticated in any way.
- **Overload**
Convenience of E-mail may result in a flood of mail.
- **Misdirection**
It is possible that you may send e-mail to an unintended recipient.
- **Junk**

Junk emails are undesirable and inappropriate emails. Junk emails are sometimes referred to as spam.

➤ **No Response**

It may be frustrating when the recipient does not read the e-mail and respond on a regular basis.

7.5.5 World Wide Web (WWW)

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with internet, enables the retrieval and display of text and media to your device.

History

Tim Berners-Lee, a British scientist, invented the World Wide Web (WWW) in 1989, while working at CERN. The Web was originally conceived and developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world. CERN is not an isolated laboratory, but rather the focal point for an extensive community that includes more than 17 000 scientists from over 100 countries. Although they typically spend some time on the CERN site, the scientists usually work at universities and national laboratories in their home countries. Reliable communication tools are therefore essential.

The basic idea of the WWW was to merge the evolving technologies of computers, data networks and hypertext into a powerful and easy to use global information system.

Difference between World Wide Web and Internet:

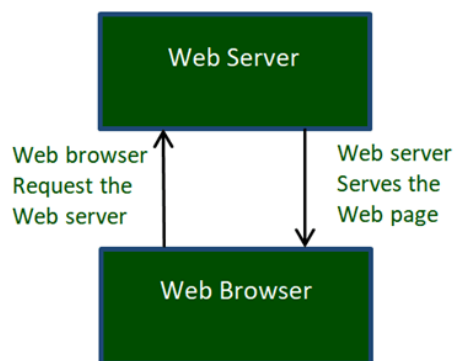
Some people use the terms 'internet' and 'World Wide Web' interchangeably. They think they are the same thing, but it is not so. Internet is entirely different from WWW. It is a worldwide network of devices like computers, laptops, tablets, etc. It enables users to send emails to other users and chat with them online. For example, when you send an email or chatting with someone online, you are using the internet.



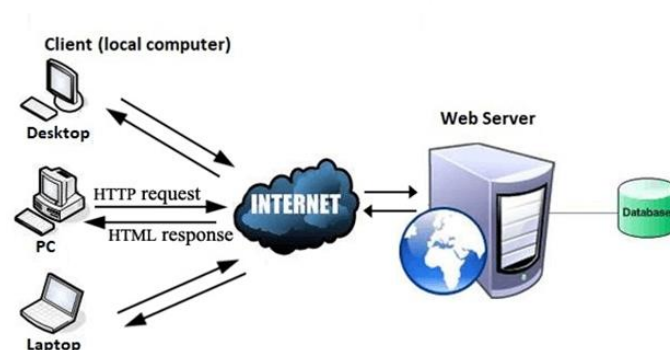
But, when you have opened a website like google.com for information, you are using the World Wide Web; a network of servers over the internet. You request a webpage from your computer using a browser, and the server renders that page to your browser. Your computer is called a client who runs a program (web browser), and asks the other computer (server) for the information it needs.

World Wide Web Working

Now, we have understood that WWW is a collection of websites connected to the internet so that people can search and share information. Now, let us understand how it works ...The Web works as per the internet's basic client-server format as shown in the following image. The servers store and transfer web pages or information to user's computers on the network when requested by the users.



A web server is a software program which serves the web pages requested by web users using a browser. The computer of a user who requests documents from a server is known as a client. Browser, which is installed on the user' computer, allows users to view the retrieved documents.



All the websites are stored in web servers. Just as someone lives on rent in a house, a website occupies a space in a server and remains stored in it. The server hosts the website whenever a user requests its WebPages, and the website owner has to pay the hosting price for the same.

The moment you open the browser and type a URL in the address bar or search something on Google, the WWW starts working. There are three main technologies involved in transferring information (web pages) from servers to clients (computers of users). These technologies include Hypertext Markup Language (HTML), Hypertext Transfer Protocol (HTTP) and Web browsers.

Hypertext Markup Language (HTML):

HTML is a standard markup language which is used for creating web pages. It describes the structure of web pages through HTML elements or tags. These tags are used to organize the pieces of content such as 'heading,' 'paragraph,' 'table,' 'Image,' and more. You don't see HTML tags when you open a webpage as browsers don't display the tags and use them only to render the content of a web page. In simple words, HTML is used to display text, images, and other resources through a Web browser.

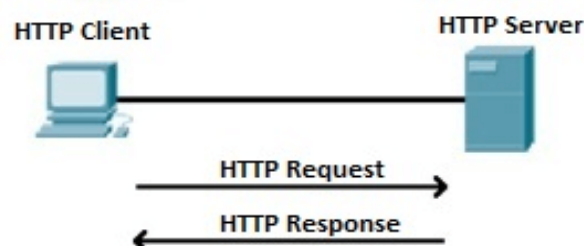
Web Browser:

A web browser, which is commonly known as a browser, is a program that displays text, data, pictures, videos, animation, and more. It provides a software interface that allows you to click hyperlinked resources on the World Wide Web. When you double click the Browser icon installed on your computer to launch it, you get connected to the World Wide Web and can search Google or type a URL into the address bar.

In the beginning, browsers were used only for browsing due to their limited potential. Today, they are more advanced; along with browsing you can use them for e-mailing, transferring multimedia files, using social media sites, and participating in online discussion groups and more. Some of the commonly used browsers include Google Chrome, Mozilla Firefox, Internet Explorer, Safari, and more.

Hypertext Transfer Protocol (HTTP):

Hyper Text Transfer Protocol (HTTP) is an application layer protocol which enables WWW to work smoothly and effectively. It is based on a client-server model. The client is a web browser which communicates with the web server which hosts the website. This protocol defines how messages are formatted and transmitted and what actions the Web Server and browser should take in response to different commands. As we enter a URL in the browser, an HTTP command is sent to the Web server, and transmits the requested Web Page.



When we open a website using a browser, a connection to the web server is opened, and the browser communicates with the server through HTTP and sends a request. HTTP is carried over TCP/IP to communicate with the server. The server processes the browser's request and sends a response, and then the connection is closed. Thus, the browser retrieves content from the server for the user.

7.5.6 Telnet

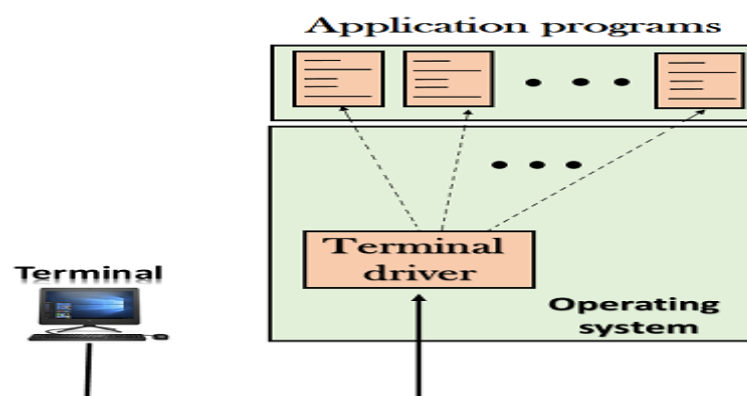
The main task of the internet is to provide services to users. For example, a user wants to run different application programs at the remote site and transfers a result to the local site. This requires a client-server program such as FTP, SMTP. But this would not allow us to create a specific program for each demand.

The better solution is to provide a general client-server program that lets the user access any application program on a remote computer. Therefore, a program that allows a user to log on to a remote computer. A popular client-server program Telnet is used to meet such demands. Telnet is an abbreviation for **Terminal Network**.

Telnet provides a connection to the remote computer in such a way that a local terminal appears to be at the remote side.

There are two types of login: Login Local and Remote login

Login Local

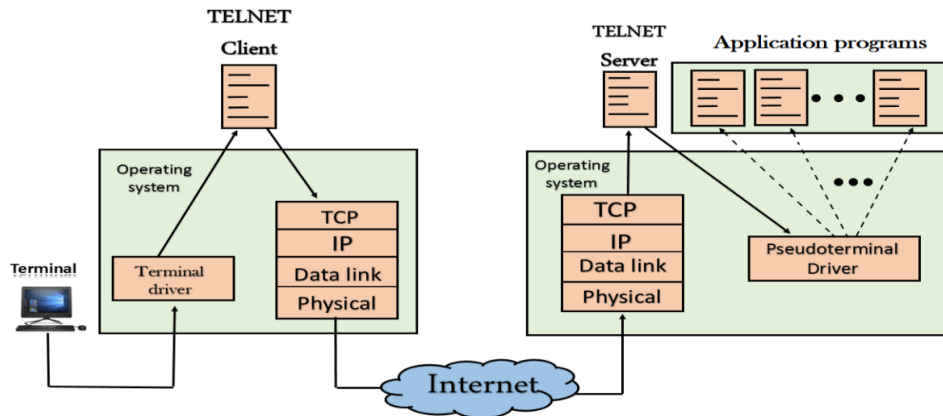


When a user logs into a local computer, then it is known as local login. When the workstation running terminal emulator, the keystrokes entered by the user are accepted by the terminal driver. The terminal driver then passes these characters to the operating system which in turn, invokes the desired application program.

However, the operating system has special meaning to special characters. For example, in UNIX some combination of characters has special meanings such as control character with "z" means suspend. Such situations do not create any problem as the terminal driver knows the meaning of such characters. But, it can cause the problems in remote login.

Remote login

When the user wants to access an application program on a remote computer, then the user must perform remote login.



How remote login occurs

At the local site

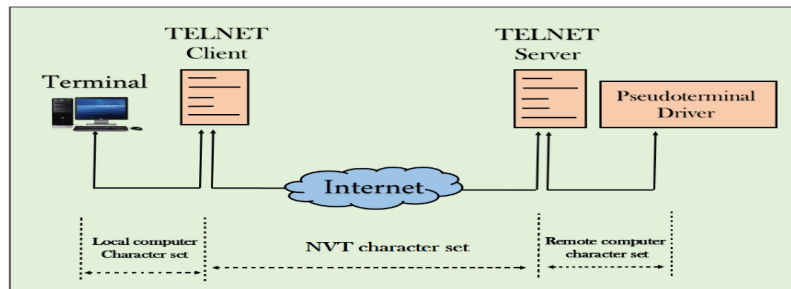
The user sends the keystrokes to the terminal driver, the characters are then sent to the TELNET client. The TELNET client which in turn, transforms the characters to a universal character set known as network virtual terminal characters and delivers them to the local TCP/IP stack

At the remote site

The commands in NVT forms are transmitted to the TCP/IP at the remote machine. Here, the characters are delivered to the operating system and then pass to the TELNET server. The TELNET server transforms the characters which can be understandable by a remote computer. However, the characters cannot be directly passed to the operating system as a remote operating system does not receive the characters from the TELNET server. Therefore it requires some piece of software that can accept the characters from the TELNET server. The operating system then passes these characters to the appropriate application program.

Network Virtual Terminal (NVT)

The network virtual terminal is an interface that defines how data and commands are sent across the network.



In today's world, systems are heterogeneous. For example, the operating system accepts a special combination of characters such as end-of-file token running a DOS operating system *ctrl+z* while the token running a UNIX operating system is *ctrl+d*.

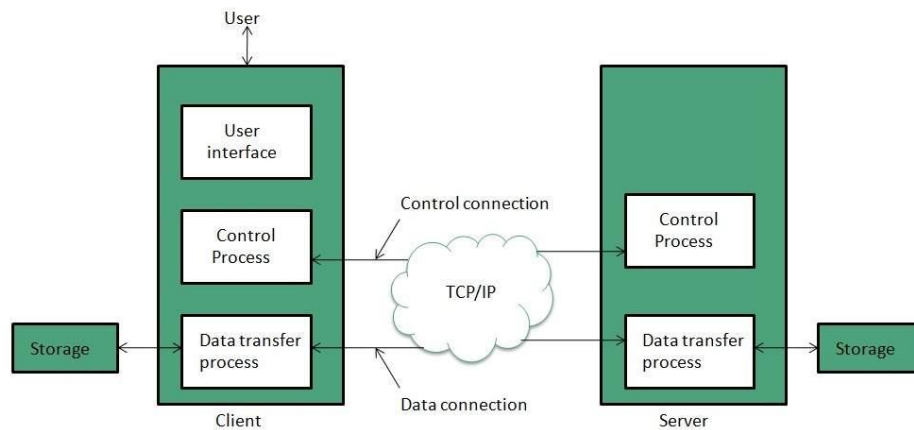
TELNET solves this issue by defining a universal interface known as network virtual interface.

The TELNET client translates the characters that come from the local terminal into NVT form and then delivers them to the network. The Telnet server then translates the data from NVT form into a form which can be understandable by a remote computer.

7.5.7 File Transfer Protocol (FTP)

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

- FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
- FTP establishes two different connections: one is for data transfer and other is for control information.
 - (a) **Data Connection:** The Data Connection uses very complex rules as data types may vary. The data connection is made between data transfer processes. The data connection opens when a command comes for transferring the files and closes when the file is transferred.
 - (b) **Control Connection:** The control connection uses very simple rules for communication. Through control connection, we can transfer a line of command or line of response at a time. The control connection is made between the control processes. The control connection remains connected during the entire interactive FTP session.
- FTP uses port 21 for the control connection and Port 20 for the data connection.



Advantages of FTP

- **Speed:** One of the biggest advantages of FTP is speed. The FTP is one of the fastest ways to transfer the files from one computer to another computer.
- **Efficient:** It is more efficient as we do not need to complete all the operations to get the entire file.
- **Security:** To access the FTP server, we need to login with the username and password. Therefore, we can say that FTP is more secure.
- **Back & forth movement:** FTP allows us to transfer the files back and forth. Suppose you are a manager of the company, you send some information to all the employees, and they all send information back on the same server.

Disadvantages of FTP

- The standard requirement of the industry is that all the FTP transmissions should be encrypted. However, not all the FTP providers are equal and not all the providers offer encryption. So, we will have to look out for the FTP providers that provides encryption.
- FTP serves two operations, i.e., to send and receive large files on a network. However, the size limit of the file is 2GB that can be sent. It also doesn't allow you to run simultaneous transfers to multiple receivers.
- Passwords and file contents are sent in clear text that allows unwanted eavesdropping. So, it is quite possible that attackers can carry out the brute force attack by trying to guess the FTP password.
- It is not compatible with every system.

7.5.8 SMTP

SMTP stands for Simple Mail Transfer Protocol. SMTP is a set of communication guidelines that allow software to transmit an electronic mail over the internet is called **Simple Mail Transfer Protocol**. It is a program used for sending messages to other computer users based on e-mail addresses.

- It provides a mail exchange between users on the same or different computers, and it also supports:
 - send a single message to one or more recipients.
 - sending message can include text, voice, video or graphics.
 - It can also send the messages on networks outside the internet.

The main purpose of SMTP is used to set up communication rules between servers. The servers have a way of identifying themselves and announcing what kind of communication they are trying to perform. They also have a way of handling the errors such as incorrect email address. For example, if the recipient address is wrong, then the receiving server replies with an error message of some kind.

SMTP Components

1. **Mail User Agent (MUA)** – Mail User Agent or User-Agent is the client application for sending and receiving emails. Ex – Gmail, Microsoft Outlook, etc.
2. **Mail Submission Agent (MSA)** – Mail Submission Agent is a server that receives mail from MUA, checks for errors, and in case of no errors passes it to the SMTP server. There exist Mail transfer agents that work as a mail submission agent too.
3. **Mail Transfer Agent (MTA)** – Mail Transfer Agent receives the email from MSA or MUA(if MSA is not used) and passes the mail(using Simple Mail Transfer Protocol SMTP) to other MTA. An MTA works in the background, while the user can't interact directly with a mail transfer agent. It will also find the MX record from the receivers' DNS zone. Ex – Postfix, SendMail etc.
4. **Mail Delivery Agents (MDA)** – Mail Delivery agents are responsible for delivering emails to recipients' mailboxes.

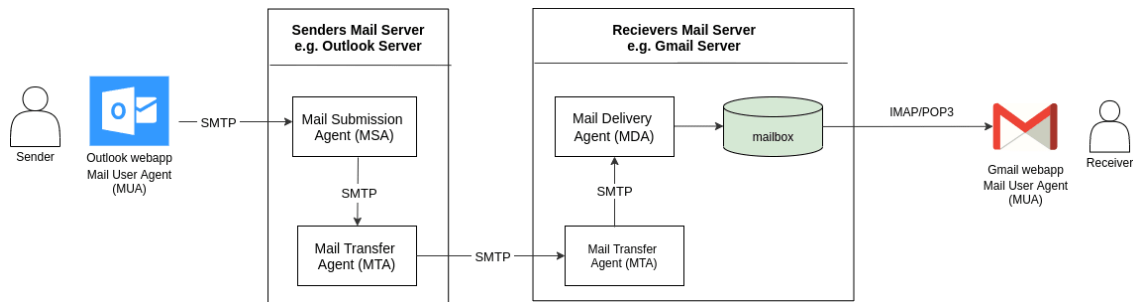
SMTP Working

An SMTP session is initiated when a client opens a connection to a server by issuing the EHLO command and the server responds with an opening message. The SMTP session consists of commands originated by the SMTP clients and the responses from the SMTP Server. A session can contain zero or more SMTP transactions. Three important commands in SMTP transaction-

1. **MAIL** – The transaction starts with the mail command. It does the sender identification.
2. **RCPT** – It gives the receiver information. If RCPT command exists before previous MAIL command the server throws 503 “Bad Sequence of Commands” response. This command can be issued multiple times.
3. **DATA** – This command tells about the beginning of a message. Once accepted the SMTP server immediately replies with 354 responses and when the entire message is

received SMTP server again replies with “**250 OK**”. SMTP indicates the end of the mail data by sending a line containing only a “.” (period or full stop).

The client who wants to send an email, the Mail client(or formally known as mail user agent or SMTP client) opens a full-duplex(two way) TCP connection with the Mail Server(SMTP Server) which is always in listening mode at the port number (25) and an SMTP session will be created. The Mail client uses TCP because it is a reliable ordered data stream channel for communication, hence the entire message will be delivered to the server without any data loss.



7. 6 REVIEW QUESTIONS

1. What is the significance of URL?
2. What is the significance of bcc and cc in email?
3. What are the two purpose of TELNET?
4. What are the similarities and differences between a domain name and its address?

7.7 SUMMARY

In this unit, we have given you a broad overview of various protocols of application layer of OSI reference model. Our discussions are focused on only some of the important protocols, but it doesn't mean that these are the only protocols associated with the application layer. Many other protocols are used by the network administrator for different purpose. You should also keep on exploring about other protocols.

7.8 REFERENCES FOR FURTHER READINGS

1. BehrouzForouzan, Data communications and Networking, Tata Mc-Graw Hill.
2. Andrew S. Tanenbaum, Computer Networks, Pearson Education.
3. William Stallings, Data and Computer Communications, Pearson education.

UNIT 8: NETWORK SECURITY

STRUCTURE

- 8.0 Objectives**
- 8.1 Introduction**
- 8.2 Common Terms**
- 8.3 Network Security Issues**
- 8.4 Goals of Network Security**
- 8.5 Types of Network Attack**
 - 8.5.1 Difference between Active and Passive Attack**
- 8.6 Firewall**
 - 8.6.1 Firewall types**
 - 8.6.2 Deployment of Firewall**
- 8.7 Virtual Private Network (VPN)**
 - 8.7.1 VPN Types**
 - 8.7.2 VPN Protocols**
- 8.6 Review Questions**
- 8.7 Summary**
- 8.8 References for Further Readings**

8.0 INTRODUCTION

The modern era is an age of information. Our life, these days are more digitally advanced than ever, and as technology improves, organization's security postures must be enhanced as well. Now, with many devices communicating with each other over wired, wireless, or cellular networks, network security is an important concept. In this unit, we will explore what are network security and its key features.

Network security is the process of taking preventative measures to protect the associating networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction or improper disclosure.

Due to the frequency and variety of existing attacks as well as the threat of new and more destructive future attacks, network security has become a central topic in the field of cyber security. By implementing and learning about network security, a small business will make their employees more responsible, a law firm will be motivated to protect its data, and an interior designer will find more effective ways to control their heavy files.

8.1 OBJECTIVES

At the end of this unit, you will be able to:

- Understand the concept of network security in data transmission.
- Discuss different terms used in network security.
- Discuss in detail the various types of Network Attacks.
- Understand the concept of firewall with its types.
- Discuss various types of Virtual Private Network (VPN) and VPN Protocols.

8.2 COMMON TERMS

Advanced Persistent Threats:

When an unauthorized user invades a network, stays for an extended period of time, and steals data without harming the network.

Adware:

Software that automatically displays or downloads material, even user is in offline mode.

Black hat:

“Black hat” is used for the hackers that break into the network to steal information that will be used to harm the owner or the users without consent. It's entirely illegal.

Bitcoin:

Bitcoin is a cryptocurrency, a form of electronic cash created by Satoshi Nakamoto.

Brute force attack:

An attacker inputs many passwords in the hope that it is eventually guessed correctly.

Breach:

The moment a hacker successfully exploits a vulnerability in a computer or device, and gains access to its files and network.

Bot/Botnet:

It is a type of software application or script that performs tasks on command, allowing an attacker to take complete control remotely of an affected computer. A collection of these infected computers is known as a “botnet” and is controlled by the hacker or “bot-herder”.

BYOD (Bring Your Own Device):

It refers to a company security policy that allows for employees’ personal devices to be used in business. A BYOD policy sets limitations and restrictions on whether or not a personal phone or laptop can be connected over the corporate network.

Catfishing:

A fake identity on a social network account is creating, usually a dating website, to target a specific victim for deception.

Cloud:

Cloud is a technology that allows us to access our files and/or services through the internet from anywhere in the world. Technically speaking, it’s a collection of computers with large storage capabilities that remotely serve requests.

Domain:

Domain is a group of computers, printers and devices that are interconnected and governed as a whole. For example, your computer is usually part of a domain at your workplace.

Deepfake:

It is an audio or video clip that has been edited and manipulated to seem real or believable. The most dangerous consequence of the popularity of deepfakes is that they can easily convince people into believing a certain story or theory that may result in user-behavior with a bigger impact as in political or financial.

DDoS:

An acronym that stands for distributed denial of service – a form of cyber attack. This attack aims to make a service such as a website unusable by “flooding” it with malicious traffic or data from multiple sources (often botnets).

Exploit:

Exploit is a malicious application or script that can be used to take advantage of a computer’s vulnerability.

Firewall:

A defensive technology designed to keep the bad guys out. Firewalls can be hardware or software-based.

Hacker:

Hacker is a person who spends time learning the details of computer programming and operating systems, how to test the limits of their capabilities, and where their vulnerabilities lie.

IP Address:

An internet version of a home address for your computer, which is identified when it communicates over a network; e.g. connecting to the internet (a network of networks).

Keylogger:

Keylogger is a computer program that records keystrokes made by a user. This user is typically unaware that their actions are being monitored and that a hacker now has access to passwords and other confidential data.

Mitigation defense:

Mitigation defense is software that doesn't stop hacking from happening, but will mitigate the effects.

Malware:

A common term that describes all forms of malicious software designed to wreak havoc on a computer. Common forms include: viruses, trojans, worms and ransomware.

Phishing:

A technique used by hackers to obtain sensitive information. For example, using hand-crafted email messages designed to trick people into divulging personal or confidential data such as passwords and bank account information.

ReCAPTCHA:

ReCAPTCHA is a service from Google that works to protect websites from spam and abuse caused by robots. A user is presented with a Turing test to distinguish them from a robot.

Ransomware:

Ransomware is a form of malware that deliberately prevents you from accessing files on your computer – holding your data hostage. It will typically encrypt files and request that a ransom be paid in order to have them decrypted or recovered.

Rootkit:

Rootkit is a kind of malware that allows cybercriminals to remotely control your computer. Rootkits are especially damaging because they are hard to detect, making it likely that this type of malware could live on your computer for a long time.

Spyware:

It is a type of malware that functions by spying on user activity without their knowledge. The capabilities include activity monitoring, collecting keystrokes, data harvesting (account information, logins, financial data), and more.

Sniffer:

A program that captures data as it travels across a network. Also called a packet sniffer.

Trialware:

Software that can only be run for a limited amount of time before it expires.

Trojan horse:

It is a computer program that appears to perform a desirable function but contains hidden code that is intended to allow unauthorized collection, modification or destruction of data.

Technical vulnerability: A flaw or bug in the hardware or software components of a system that leaves it vulnerable to security breach.

Virtual Private Network (VPN):

VPN is a tool that allows the user to remain anonymous while using the internet by masking the location and encrypting traffic.

Virus:

A type of malware aimed to corrupt, erase or modify information on a computer before spreading to others.

Vulnerability:

A weakness in the hardware or software or security plan that leaves a system or network open to threat of unauthorized access or damage or destruction of data.

Worm:

A piece of malware that can replicate itself in order to spread the infection to other connected computers.

White hat:

It breaches the network to gain sensitive information with the owner's consent – making it completely legal. This method is usually employed to test infrastructure vulnerabilities.

8.3 NETWORK SECURITY ISSUES

The importance of network security in the modern business atmosphere has increased after a major portion of the workforce went remote due to COVID-19. Today, the office system is distributed in one huge network across multiple geographical locations. A centralized system

to protect network devices from being breached doesn't exist in the same capacity. This scenario leaves more vulnerable points that hackers can take advantage of.

In today's business infrastructure, network security is not limited to IT professionals and companies connected with it. Network security is for everyone — lawyers, interior decorators, musicians, investment bankers, etc., will all find a network security system beneficial for their work and business.

A *network attack* can be defined as any method, process, or means used to maliciously attempt to compromise network security. Network security is the process of preventing network attacks across a given network, but the techniques and methods used by the attacker further distinguish whether the attack is an active cyber attack, a passive type attack, or some combination of the two.

8.4 GOALS OF NETWORK SECURITY

As discussed in earlier sections, there exists large number of vulnerabilities in the network. Thus, during transmission, data is highly vulnerable to attacks. An attacker can target the communication channel, obtain the data, and read the same or re-insert a false message to achieve his nefarious aims.

Network security is not only concerned about the security of the computers at each end of the communication chain; however, it aims to ensure that the entire network is secure.

Network security entails protecting the usability, reliability, integrity, and safety of network and data. Effective network security defeats a variety of threats from entering or spreading on a network.

The primary goals of network security are Confidentiality, Integrity, and Availability. These three pillars of Network Security are often represented as CIA triangle.

- **Confidentiality:** The function of confidentiality is to protect precious business data from unauthorized persons. Confidentiality part of network security makes sure that the data is available only to the intended and authorized persons.
- **Integrity:** This goal means maintaining and assuring the accuracy and consistency of data. The function of integrity is to make sure that the data is reliable and is not changed by unauthorized persons.
- **Availability:** The function of availability in Network Security is to make sure that the data, network resources/services are continuously available to the legitimate users, whenever they require it.

Vulnerabilities & Attacks

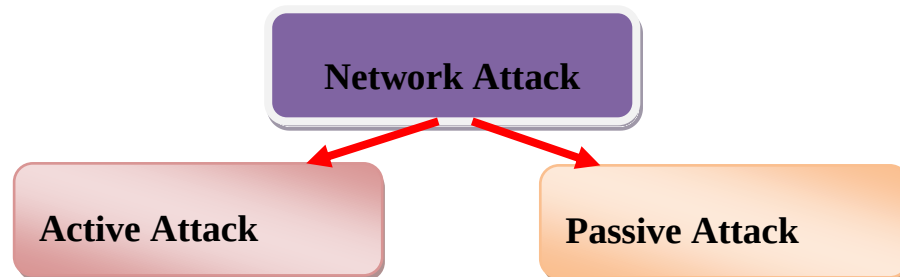
The common vulnerability that exists in both wired and wireless networks is an “unauthorized access” to a network. An attacker can connect his device to a network through unsecure hub/switch port. In this regard, wireless network are considered less secure than

wired network, because wireless network can be easily accessed without any physical connection.

After accessing, an attacker can exploit this vulnerability to launch attacks such as under:

- Sniffing the packet data to steal valuable information.
- Denial of service to legitimate users on a network by flooding the network medium with spurious packets.
- Spoofing physical identities (MAC) of legitimate hosts and then stealing data or further launching a 'man-in-the-middle' attack.

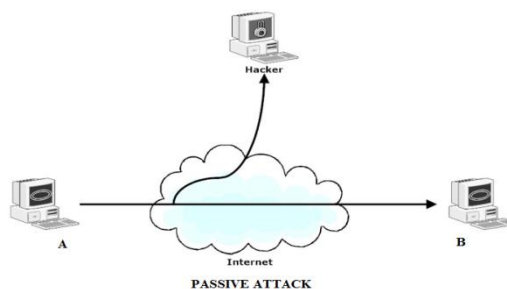
8.5 TYPES OF NETWORK ATTACK



Passive Attack:

In passive attacks, hackers monitor and scan systems for vulnerabilities or entry points that allow them to intercept information without changing any of it. In a passive attack, an intruder monitors a system and network communications and scans for open ports and other vulnerabilities. For example, they might exploit an unpatched system or take advantage of an expired certificate on a security device.

Once the intruder has infiltrated the network, they can collect information in a couple of ways. In a footprinting passive attack, the intruder will try to collect as much intelligence as they can to use it later to attack the target system or network in a later step. An example is when an intruder records network traffic using a packet analyzer tool, such as Wireshark for later analysis. Installing a keylogger is another sort of passive attack, where an intruder waits for the user to enter their credentials and records them for later use.



The two most common use cases of passive attacks are:

1. **Traffic analysis:** In this type, an attacker monitors communication channels to collect a range of information, including human and machine identities, locations of these identities and types of encryption used, if applicable.
2. **Release of message contents:** In this type, an attacker will monitor an unprotected communication medium—like unencrypted email or telephone call—and intercept it for sensitive information.

Other types of passive attacks include “passive reconnaissance,” where an attacker tries to gain important information about the target organization connected to the internet without sending any traffic (packets) to the target server or network. An example of such type of attack includes browsing a website contents for relevant information (such as employee contact information) that can be used in active attacks or finding files that have been left unprotected on a target server, such as meeting papers or intellectual property.

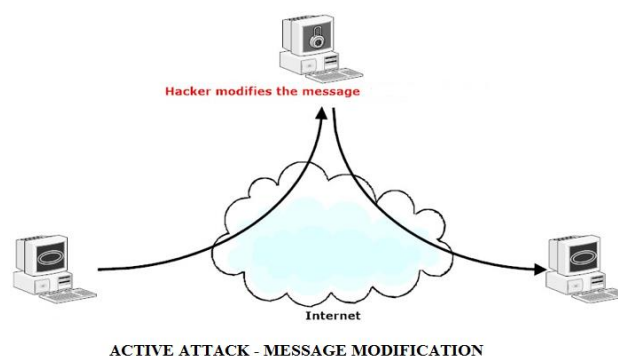
Detecting a passive attack is very difficult and impossible in many cases because it does not involve data alteration in any way. However, you can implement protective measures to stop it, including:

By using encryption techniques to scramble messages, making them unreadable for any unintended recipients.

Active Attack:

In an active attack, hackers attempt to modify the integrity and availability of the information they have intercepted—with the goal of gaining access or greater privileges. Simply put, hackers may use data they have gathered during passive attacks to compromise a target in an active attack.

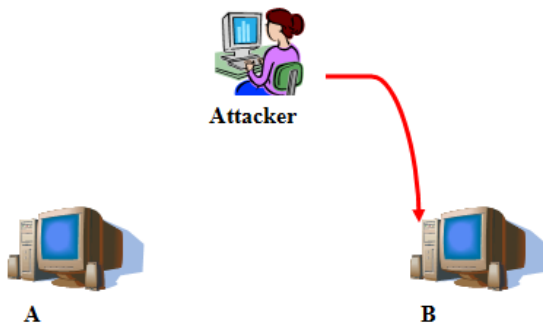
The attacker can read and update the data without the information of any of the users. In Active Attack, the attacker tries to induce noise in the data transmission.



Active attack examples

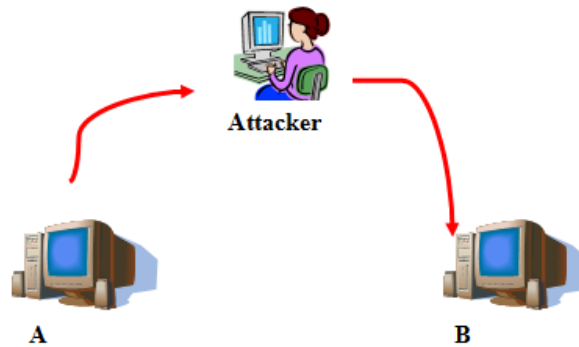
1) Masquerade

Assume that A and B are connected and they are transferring data to each other. A and B are genuine users. In the Masquerade attack, the attacker used the identity of the authentic users and he breaks into the communication and behaves like the authentic user and grabs all the data.



2) Relay:

Assume that A and B are connected and they are transferring data to each other. A is sending some message to B. The message is on its way but in between the attacker captures the message and now not only he can read the message but he can update and modify it too. He can create error bits in the message. Error bits are the bits that don't belong to the original message.



3) Denial of service:

In this attack, the attacker sends a lot of requests to the server to increase the traffic. If the server has a lot of requests then it will take a lot of time to respond to the genuine requests which are made by the authentic users. In this way, by increasing the traffic on the server, he can slow down the server. In this way, the authentic users will not get a response from the server. In this way, their service is denied.

Types of active attacks include:

- Denial of service (DoS)
- Distributed Denial of Service (DDoS)
- Session replay
- Masquerade
- Message modification
- Trojans

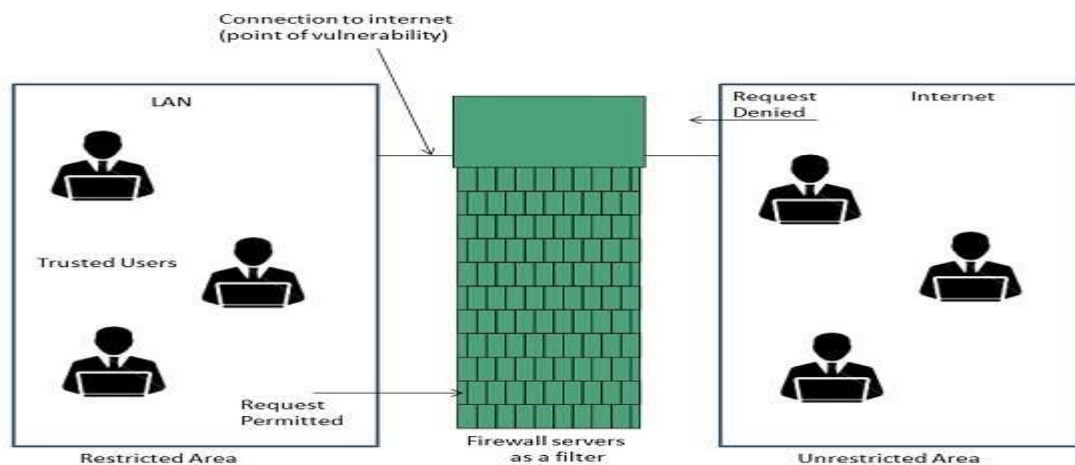
7.5.1 Difference between Active and Passive Attack

Active Attack	Passive Attack
Attacker needs to have physical control of the media or network.	Attacker merely needs to observe the communication in the media or network.
It can be easily detected.	It cannot be easily detected.
It affects the system.	It does not affect the system.
It involves a modification of data.	It involves the monitoring of data.
Types of active attacks are Masquerade, session replay, denial of service, distributed denial of service.	Types of passive attacks are the Release of a message, traffic analysis.
It does not check for loopholes or vulnerabilities.	It scans the ports and network in the search of loopholes and vulnerabilities.
It is difficult to prevent network from active attack.	Passive attacks can be prevented.

8.6 FIREWALL

Firewall is a filter between Local Area Network (LAN) and the Internet. It allows keeping private resources confidential and maximizes the security. It controls network traffic, in both directions (to and from).

The following diagram depicts a sample firewall between LAN and the internet. The connection between the two is the point of vulnerability. Both hardware and the software can be used at this point to filter network traffic.



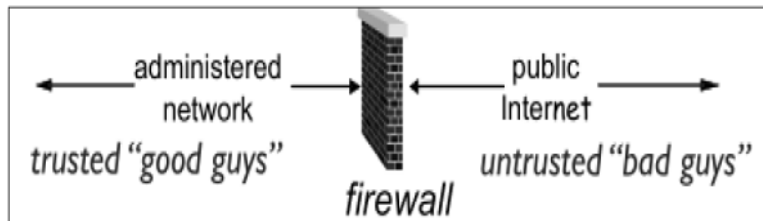
There are two types of Firewall system:

One works by using filters at the network layer and

The other works by using proxy servers at the user, application, or network layer.

Firewall management must be configured by both system managers and the network managers. The amount of filtering a firewall varies; the amount of filtering may be different in different directions for same firewall.

Firewall is a network device with inbuilt software that isolates organization's internal network from larger outside network/Internet. It can be hardware, software, or combined system that prevents unauthorized access to or from internal network. All data packets entering or leaving the internal network pass through the firewall, which examines each packet and blocks those that do not meet the predefined security criteria.



Deploying firewall at network boundary is like aggregating the security at a single point. It is analogous to deploying a security guard at the entrance in an apartment and not necessarily at each flat.

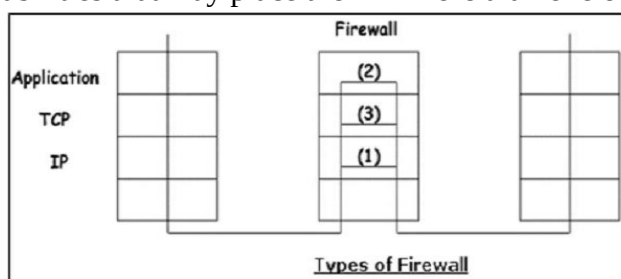
Firewall is considered as an essential to achieve network security for the following reasons –

1. Internal network and hosts are unlikely to be properly secured.
2. Internet is a dangerous place with criminals, users from competing companies, disgruntled ex-employees, spies from unfriendly countries, vandals, etc.
3. To prevent illegal modification/access to internal data by an outsider attacker.
4. To prevent an attacker from launching denial of service (DoS) attacks on network resource.

8.6.1 Firewall types

1. Packet filter (Stateless & Stateful)
2. Application-level gateway
3. Circuit-level gateway

These three categories, however, are not mutually exclusive. Modern firewalls have a mix of abilities that may place them in more than one of the three categories.



1. Packet Filtering Firewall

In this type of firewall deployment, the internal network is connected to the external network/Internet via a router firewall. The firewall filters data packet-by-packet. Packet filtering firewalls allow or block the packets mostly based on criteria such as source and/or

destination IP addresses, protocol, source and/or destination port numbers, and various other parameters within the IP header.

The decision can be based on factors other than IP header fields such as ICMP message type, TCP SYN and ACK bits, etc. Packet filter rule has two parts –

- **Selection criteria** – It is used as a condition and pattern matching for decision making.
- **Action field** – This part specifies action to be taken if an IP packet meets the selection criteria. The action could be either block (deny) or permit (allow) the packet across the firewall.

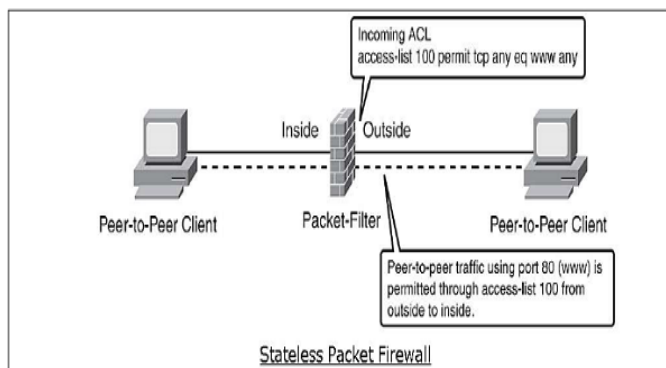
Packet filtering is generally accomplished by configuring Access Control Lists (ACL) on routers or switches. ACL is a table of packet filter rules.

As traffic enters or exits an interface, firewall applies ACLs from top to bottom to each incoming packet, finds matching criteria and either permits or denies the individual packets.

Stateless firewall

It is a kind of a rigid tool. It looks at packet and allows it if it meets the criteria even if it is not part of any established ongoing communication.

Hence, such firewalls are replaced by stateful firewalls in modern networks. This type of firewalls offer a more in-depth inspection method over the only ACL based packet inspection methods of stateless firewalls.



Stateful firewall

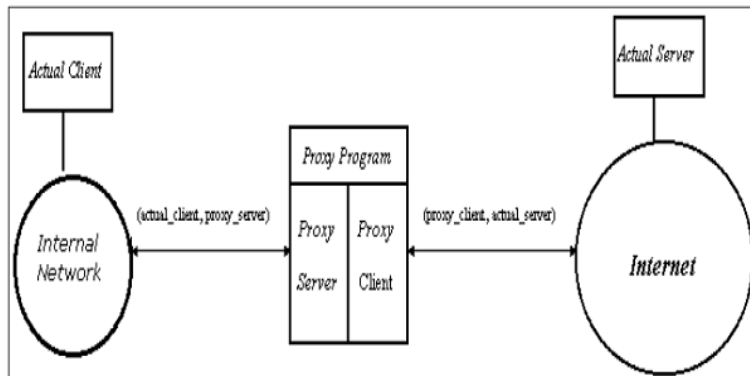
It monitors the connection setup and teardown process to keep a check on connections at the TCP/IP level. This allows them to keep track of connections state and determine which hosts have open, authorized connections at any given point in time.

They reference the rule base only when a new connection is requested. Packets belonging to existing connections are compared to the firewall's state table of open connections, and decision to allow or block is taken. This process saves time and provides added security as well. No packet is allowed to trespass the firewall unless it belongs to already established connection. It can timeout inactive connections at firewall after which it no longer admit packets for that connection.

2. Application Gateways:

An application-level gateway acts as a relay node for the application-level traffic. They intercept incoming and outgoing packets, run proxies that copy and forward information across the gateway, and function as a proxy server, preventing any direct connection between a trusted server or client and an untrusted host.

The proxies are application specific. They can filter packets at the application layer of the OSI model.



An application-specific proxy accepts packets generated by only specified application for which they are designed to copy, forward, and filter. For example, only a Telnet proxy can copy, forward, and filter Telnet traffic.

If a network relies only on an application-level gateway, incoming and outgoing packets cannot access services that have no proxies configured. For example, if a gateway runs FTP and Telnet proxies, only packets generated by these services can pass through the firewall, and all other services are blocked.

Application-level Filtering

An application-level proxy gateway, examines and filters individual packets, rather than simply copying them and blindly forwarding them across the gateway. Application-specific proxies check each packet that passes through the gateway, verifying the contents of the packet up through the application layer. These proxies can filter particular kinds of commands or information in the application protocols.

Application gateways can restrict specific actions from being performed. For example, the gateway could be configured to prevent users from performing the 'FTP put' command. This can prevent modification of the information stored on the server by an attacker.

Transparent

Although application-level gateways can be transparent, many implementations require user authentication before users can access an untrusted network, a process that reduces true transparency. Authentication may be different if the user is from the internal network or from the Internet. For an internal network, a simple list of IP addresses can be allowed to connect to external applications. But from the Internet side a strong authentication should be implemented.

An application gateway actually relays TCP segments between the two TCP connections in the two directions (Client ↔ Proxy ↔ Server).

For outbound packets, the gateway may replace the source IP address by its own IP address. The process is referred to as **Network Address Translation** (NAT). It ensures that internal IP addresses are not exposed to the Internet.

3. Circuit-Level Gateway:

The circuit-level gateway is an intermediate solution between the packet filter and the application gateway. It runs at the transport layer and hence can act as proxy for any application.

Similar to an application gateway, the circuit-level gateway also does not permit an end-to-end TCP connection across the gateway. It sets up two TCP connections and relays the TCP segments from one network to the other. But it does not examine the application data like application gateway. Hence, sometime it is called as ‘Pipe Proxy’.

SOCKS

SOCKS (RFC 1928) refer to a circuit-level gateway. OpenText™ SOCKS Client is a Windows–certified security solution that connects to hosts across firewalls. It is a networking proxy mechanism that enables hosts on one side of a SOCKS server to gain full access to hosts on the other side without requiring direct IP reachability. The client connects to the SOCKS server at the firewall. Then the client enters a negotiation for the authentication method to be used, and authenticates with the chosen method.

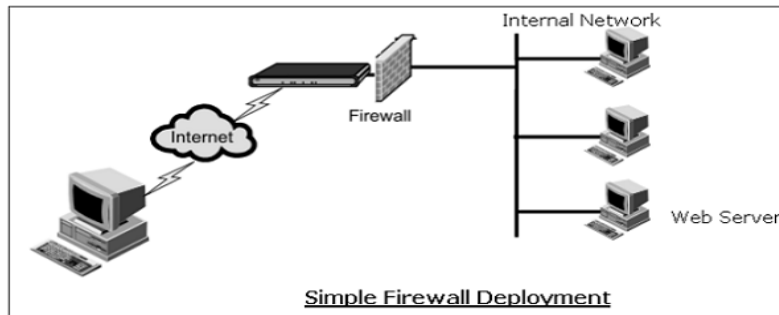
The client sends a connection relay request to the SOCKS server, containing the desired destination IP address and transport port. The server accepts the request after checking that the client meets the basic filtering criteria. Then, on behalf of the client, the gateway opens a connection to the requested untrusted host and then closely monitors the TCP handshaking that follows.

The SOCKS server informs the client, and in case of success, starts relaying the data between the two connections. Circuit level gateways are used when the organization trusts the internal users, and does not want to inspect the contents or application data sent on the Internet.

8.6.2 Deployment of Firewall

A firewall is a mechanism used to control network traffic ‘into’ and ‘out’ of an organizational internal network. In most cases these systems have two network interfaces, one for the external network such as the Internet and the other for the internal side.

The firewall process can tightly control what is allowed to traverse from one side to the other. An organization that wishes to provide external access to its web server can restrict all traffic arriving at firewall expect for port 80 (the standard http port). All other traffic such as mail traffic, FTP, SNMP, etc., is not allowed across the firewall into the internal network. An example of a simple firewall is shown in the following diagram.

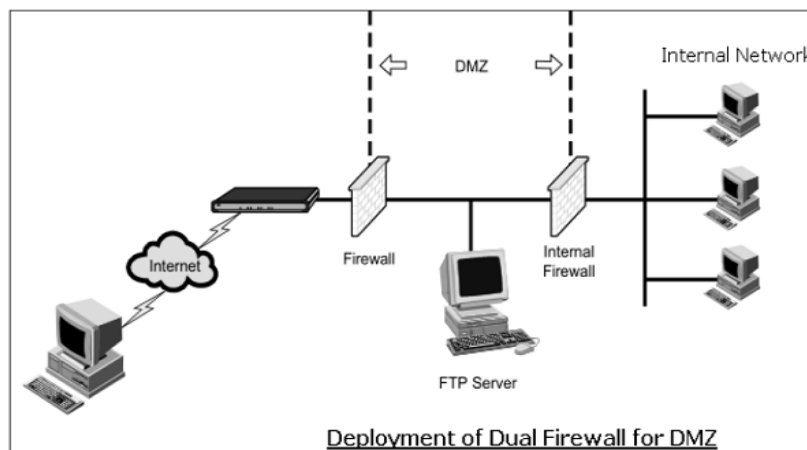


In the above simple deployment, though all other accesses from outside are blocked, it is possible for an attacker to contact not only a web server but any other host on internal network that has left port 80 open by accident or otherwise.

Hence, the problem most organizations face is how to enable legitimate access to public services such as web, FTP, and e-mail while maintaining tight security of the internal network. The typical approach is deploying firewalls to provide a Demilitarized Zone (DMZ) in the network.

In this setup (illustrated in following diagram), two firewalls are deployed; one between the external network and the DMZ, and another between the DMZ and the internal network. All public servers are placed in the DMZ.

With this setup, it is possible to have firewall rules which allow public access to the public servers but the interior firewall can restrict all incoming connections. By having the DMZ, the public servers are provided with adequate protection instead of placing them directly on external network.



Intrusion Detection / Prevention System

The packet filtering firewalls operate based on rules involving TCP/UDP/IP headers only. They do not attempt to establish correlation checks among different sessions.

Intrusion Detection/Prevention System (IDS/IPS) carry out Deep Packet Inspection (DPI) by looking at the packet contents. For example, checking character strings in packet against database of known virus, attack strings.

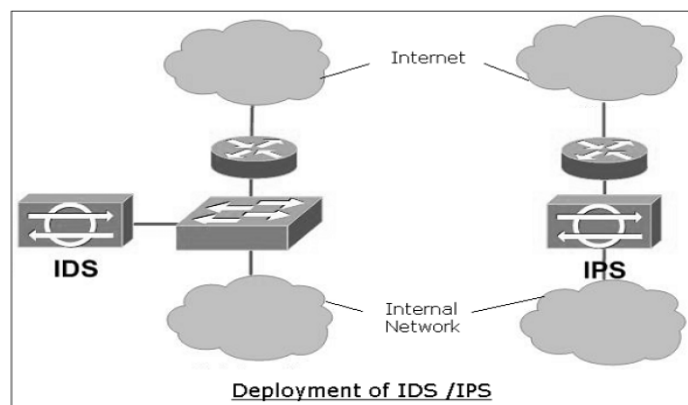
Application gateways do look at the packet contents but only for specific applications. They do not look for suspicious data in the packet. IDS/IPS looks for suspicious data contained in packets and tries to examine correlation among multiple packets to identify any attacks such as port scanning, network mapping, and denial of service and so on.

Difference between IDS and IPS

IDS and IPS are similar in detection of anomalies in the network. IDS is a ‘visibility’ tool whereas IPS is considered as a ‘control’ tool.

IPS sits off to the side of the network, monitoring traffic at many different points, and provide visibility into the security state of the network. In case of reporting of anomaly by IDS, the corrective actions are initiated by the network administrator or other device on the network.

IPS are like firewall and they sit in-line between two networks and control the traffic going through them. It enforces a specified policy on detection of anomaly in the network traffic. Generally, it drops all packets and blocks the entire network traffic on noticing an anomaly till such time an anomaly is addressed by the administrator.



8.7 VIRTUAL PRIVATE NETWORK (VPN)

When you use a private WAN connection from a service provider, you trust them that they treat your data confidential. The service provider will make sure that they separate traffic from different customers and that nobody else is able to see your data. When you send traffic from A to B using the Internet, you have no control at all which networks are used to get from the source to the destination. Someone in between the traffic path might be capturing your packets and you wouldn't know.

VPNs (Virtual Private Network) help by establishing a secure connection over an insecure network, such as the Internet. This is a great alternative to private WAN connections since Internet access is usually cheaper and it's available pretty much everywhere.

VPNs provide a couple of features such as:

- 🚦 **Confidentiality:** preventing anyone from reading your data. This is implemented with encryption.

- ✚ **Authentication:** verifying that the router/firewall or remote user that is sending VPN traffic is a legitimate device or router.
- ✚ **Integrity:** verifying that the VPN packet wasn't changed somehow during transit.
- ✚ **Anti-replay:** preventing someone from capturing traffic and resending it, trying to appear as a legitimate device/user.

8.7.1 VPN Types

There are two common VPN types that we use:

- **site-to-site VPN**
- **client-to-site VPN (remote user)**

Site-to-site VPN

With the site-to-site VPN, we have a network device at each site, between these two network devices we build a VPN tunnel. Each end of the VPN tunnel will encrypt the original IP packet, adds a VPN header, a new IP header and then forwards the encrypted packet to the other end of the tunnel.

Client-to-site VPN

The client-to-site VPN is also called the **remote user VPN**. The user installs a VPN client on his/her computer, laptop, smartphone or tablet. The VPN tunnel is established between the user's device and the remote network device.

8.7.2 VPN Protocols

There are a number of VPN protocols, the most common ones are:

- **IPsec**
- **PPTP**
- **L2TP**
- **SSL VPN**

IPsec

The IP protocol itself doesn't have any security features at all, which is why IPsec was created. IPsec framework is responsible for confidentiality, integrity, authentication and anti-replay features on network layer of the OSI model. It uses a variety of protocols and the advantage of a framework is that the protocols it uses can change in the future. For example, encryption algorithms like DES, 3DES or AES are presently most favourable protocols but if a new algorithm is created, IPsec might use it in the future like:-

- ✓ Creating a site-to-site VPN tunnel.
- ✓ Creating a client-to-site (remote user) VPN tunnel.
- ✓ Between two servers to authenticate and/or encrypt traffic.

PPTP

PPTP (Point to Point Tunnelling Protocol) is one of the older VPN protocols, released in 1995. It uses a GRE tunnel for tunnelling and PPP for authentication. Encryption is done with the MPPE protocol. PPTP is supported on many clients and operating systems. Although, PPTP has been proven insecure, therefore this protocol is not used anymore.

L2TP

L2TP (Layer Two Tunnelling Protocol) is an extension of PPTP. L2TP can be used to “bridge” two remote LANs together and you want to use a single subnet on both sites. L2TP itself does not offer any encryption or anything, therefore use it together with IPsec and together, it’s often referred to as L2TP/IPsec

SSL VPN

SSL (Secure Sockets Layer) is a protocol that is normally used to encrypt data traffic between a web browser and web server. Even though it’s called SSL VPN, but nowadays network engineers use TLS (Transport Layer Security) for HTTPS, which is the successor of SSL.

One of the advantages of SSL VPN is that since it uses HTTPS, it’s more acceptable.

Most public wifi hotspots do permit HTTPS traffic while some might block other traffic like IPsec. Most SSL VPN solutions offer a “portal” through the web browser to access applications. For some advanced features, a software client has to install.

8.6 REVIEW QUESTIONS

1. What are the potential outcomes of a network security attack?
2. What are the basic objectives of network security?
3. What is the importance of firewalls in network security?
4. What is the significance of Virtual Private Network?
5. Why there are different protocols associated with VPN?

8.7 SUMMARY

Therefore, you can notice that you have to cover a broad range of topics for network security. The network security topics presented here provide you an interaction with network security issues. Most important of all, the above-mentioned questions also reflect on the fundamentals of various aspects of network security. So, aspiring candidates should start their preparations immediately for network security for further reading and explore the same from the reference books mentioned in the next section here.

REFERENCES FOR FURTHER READINGS

1. Behrouz Forouzan, Data communications and Networking, Tata Mc-Graw Hill.
2. Andrew S. Tanenbaum, Computer Networks, Pearson Education.
3. William Stallings, Data and Computer Communications, Pearson education.



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

**The Motto of the University
(SEWA)**

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



**Bachelor of Computer Applications (BCA)
Course Name: Object Oriented Programming
Course Code: BCA-4-03T**

**ADDRESS: C/28, THE LOWER MALL, PATIALA-147001
WEBSITE: www.psou.ac.in**



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME COORDINATOR :

Dr. Monika Pathak

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

PROGRAMME CO-COORDINATOR :

Dr. Gaurav Dhiman

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

COURSE COORDINATOR :

Dr. Karan Sukhija

Assistant Professor, School of Sciences and Emerging Technologies
JGND PSOU, Patiala



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

BCA-4-03T: Object Oriented Programming

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 40%

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Course: Object Oriented Programming	
Course Code: BCA-4-03T	
Course Outcomes (COs) After the completion of this course, the students will be able to:	
CO1	Develop understanding of writing object-oriented programs that combine functions and data.
CO2	Gain a thorough understanding of the core principles of OOP, including encapsulation, inheritance, and polymorphism.
CO3	Learn how to apply OOP concepts to solve programming problems, design software systems, and develop reusable code.
CO4	Understand how to create classes and objects in a programming language that supports OOP
CO5	Learn how to use inheritance to create hierarchies of classes and reuse code efficiently.

Detailed Contents:

Module	Module Name	Module Contents
Section-A		
Module I	Introduction to OOP	Introduction to OOP: • Basic concepts (objects, classes, inheritance, polymorphism, encapsulation) • Advantages of OOP over procedural programming Classes and Objects: • Declaring classes • Creating objects • Access specifiers (public, private, protected) • Constructors and destructors • Static members
Module II	Inheritance and Polymorphism	Inheritance: • Base and derived classes • Types of inheritance (single, multiple, multilevel, hierarchical) • Access control in inheritance Polymorphism: • Function overloading • Operator overloading • Virtual functions and runtime polymorphism • Abstract classes and pure virtual functions
Section-B		
Module III	Encapsulation and Interfaces and Abstract Classes	Encapsulation: • Data hiding • Accessor and mutator methods • Benefits of encapsulation Interfaces and Abstract Classes: • Declaring interfaces • Implementing interfaces • Abstract classes and methods
Module IV	Exception Handling and File Handling	Exception Handling: • Handling exceptions using try-catch blocks • Throwing exceptions • Custom exceptions File Handling: • Reading from and writing to files • File streams (File Input Stream, File Output Stream, etc.)

Reference Books:

- Herbert Schildt, “The Complete Reference C++”, Tata McGraw-Hill.
- Deitel and Deitel, “C++ How to Program”, Pearson Education.
- Robert Lafore, “Object Oriented Programming in C++”, Galgotia Publications.
- Bjarne Strastrup, “The C++ Programming Language”, Addison-Wesley Publication Co.
- Stanley B. Lippman, Josee Lajoie, “C++ Primer”, Pearson Education.
- E. Balagurusamy, “Object Oriented Programming with C++”, Tata McGraw-Hill

Section A

Module I

Introduction to OOP:

OOP stands for Object-Oriented Programming. Object-Oriented Programming & System (OOPS) concepts in Java helps reduce code complexity and enables the reusability of code. Programmers feel like working with real-life entities or objects. Object-oriented programming is a programming paradigm that brings together data and methods in a single entity called object. This promotes greater understanding as well as flexibility and maintenance of code over a long period of time. The basic concepts about oops in java are given below:

Basic Concepts of oops:

Objects & Classes:

Objects are the basic unit of OOPS representing real-life entities. They are invoked with the help of methods. These methods are declared within a class. Usually, a new keyword is used to create an object of a class in Java. Class is a predefined or user-defined template from which objects are created. It represents properties/methods that are common to all objects of the same class. It has several features, such as access modifiers, class names, interfaces, and class bodies.

Abstraction

Abstraction means showing only the relevant details to the end-user and hiding the irrelevant features that serve as a distraction. For example, during an ATM operation, we only answer a series of questions to process the transaction without any knowledge about what happens in the background between the bank and the ATM.

Example Program of Abstraction in Java

```
abstract class Bike
{
    Bike()
    {
        System.out.println("The Street Bob. ");
    }
    abstract void drive();
    void weight()
    {
        System.out.println("Light on its feet with a hefty : 630 lbs.");
    }
}
class HarleyDavidson extends Bike
```

```

{
    void drive()
    {
        System.out.println("Old-school yet relevant.");
    }
}
public class Abstraction
{
    public static void main (String args[])
    {
        Bike obj = new HarleyDavidson();
        obj.drive();
        obj.weight();
    }
}

```

Output:

```

The Street Bob.
Old-school yet relevant.
Light on its feet with a hefty: 630 lbs.

```

3. Encapsulation

Encapsulation is a means of binding data variables and methods together in a class. Only objects of the class can then be allowed to access these entities. This is known as data hiding and helps in the insulation of data.

Example Program of Encapsulation in Java

```

class Encapsulate
{
    private String Name;
    private int Height;
    private int Weight;
    public int getHeight()
    {
        return Height;
    }
    public String getName()
    {
        return Name;
    }
    public int getWeight()
    {
        return Weight;
    }

    public void setWeight( int newWeight)
    {
        Weight = newWeight;
    }
    public void setName(String newName)

```

```

    {
        Name = newName;
    }
    public void setHeight( int newHeight)
    {
        Height = newHeight;
    }
}
public class TestEncapsulation
{
    public static void main (String[] args)
    {
        Encapsulate obj = new Encapsulate ();
        obj.setName("Abi");
        obj.setWeight(70);
        obj.setHeight(178);
        System.out.println("My name: " + obj.getName());
        System.out.println("My height: " + obj.getWeight());
        System.out.println("My weight " + obj.getHeight());
    }
}

```

Output:

```

    My name: Abi
    My height: 70
    My weight: 178

```

4. Inheritance – Single, Multilevel, Hierarchical, and Multiple

Inheritance is the process by which one class inherits the functions and properties of another class. The main function of inheritance is the reusability of code. Each subclass only has to define its features. The rest of the features can be derived directly from the parent class.

Single Inheritance – Refers to a parent-child relationship where a child class extends the parent class features. Class Y extends Class X.

Multilevel Inheritance – Refers to a parent-child relationship where a child class extends another child's class. Class Y extends Class X. Class Z extends Class Y.

Hierarchical Inheritance – This refers to a parent-child relationship where several child classes extend one class. Class Y extends Class X, and Class Z extends Class X.

Multiple Inheritance – Refers to a parent-child relationship where one child class is extending from two or more parent classes. JAVA does not support this inheritance.

Example Program of Inheritance in Java

```

class Animal
{
    void habit()
    {
        System.out.println("I am nocturnal!! ");
    }
}

```

```

class Mammal extends Animal
{
    void nature()
    {
        System.out.println("I hang upside down!! ");
    }
}
class Bat extends Mammal
{
    void hobby()
    {
        System.out.println("I fly !! ");
    }
}
public class Inheritance
{
    public static void main(String args[])
    {
        Bat b = new Bat();
        b.habit();
        b.nature();
        b.hobby();
    }
}

```

Output:

```

I am nocturnal!!
I hang upside down!!
I fly !!

```

5. Polymorphism – Static and Dynamic

It is an object-oriented approach that allows the developer to assign and perform several actions using a single function. For example, “+” can be used for addition as well as string concatenation. Static Polymorphism is based on Method Overloading, and Dynamic Polymorphism is based on Method Overriding.

Example Program of Static Polymorphism with Method Overloading

```

Method Overloading
class CubeArea
{
    double area(int x)
    {
        return 6 * x * x;
    }
}
class SphereArea
{
    double area(int x)
    {

```

```

        return 4 * 3.14 * x * x;
    }
}
class CylinderArea
{
    double area(int x, int y)
    {
        return x * y;
    }
}
public class Overloading
{
    public static void main(String []args)
    {
        CubeArea ca = new CubeArea();
        SphereArea sa = new SphereArea();
        CylinderArea cia = new CylinderArea();
        System.out.println("Surface area of cube = "+ ca.area(1));
        System.out.println("Surface area of sphere= "+ sa.area(2));
        System.out.println("Surface area of cylinder= "+ cia.area(3,4));
    }
}

```

Output:

```

Surface area of cube = 6.0
Surface area of sphere= 50.24
Surface area of cylinder= 12.0

```

Example Program of Dynamic Polymorphism with Method Overriding

```

class Shape
{
    void draw()
    {
        System.out.println("Your favorite shape");
    }
    void numberOfSides()
    {
        System.out.println("side = 0");
    }
}
class Square extends Shape
{
    void draw()
    {
        System.out.println("SQUARE ");
    }
    void numberOfSides()
    {
        System.out.println("side = 4 ");
    }
}

```

```

class Pentagon extends Shape
{
    void draw()
    {
        System.out.println("PENTAGON ");
    }
    void numberOfSides()
    {
        System.out.println("side= 5");
    }
}
class Hexagon extends Shape
{
    void draw()
    {
        System.out.println("HEXAGON ");
    }
    void numberOfSides()
    {
        System.out.println("side = 6 ");
    }
}
public class Overriding
{
    public static void main(String []args)
    {
        Square s = new Square();
        s.draw();
        s.numberOfSides();
        Pentagon p = new Pentagon();
        p.draw();
        p.numberOfSides();
        Hexagon h = new Hexagon();
        h.draw();
        h.numberOfSides();
    }
}

```

Output:

```

    SQUARE;
    side = 4;
    PENTAGON;
    side= 5
    HEXAGON
    side = 6

```

Advantages of OOP over procedural programming

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods. Object-oriented programming has several advantages over procedural

programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time.

Classes and Objects:

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake. A Class is like an object constructor, or a "blueprint" for creating objects.

Java Classes

A class in Java is a set of objects which shares common characteristics/ behavior and common properties/ attributes. It is a user-defined blueprint or prototype from which objects are created. For example, Student is a class while a particular student named Ravi is an object.

Properties of Java Classes

1. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
2. Class does not occupy memory.
3. Class is a group of variables of different data types and a group of methods.
4. A Class in Java can contain:
 - Data member
 - Method
 - Constructor
 - Nested Class
 - Interface

Class Declaration in Java:

```
access_modifier class <class_name>
{
    data member;
    method;
    constructor;
    nested class;
```



```

    interface;
}
Example:
// Java Program for class example
class Student
{
    // data member (also instance variable)
    int id;
    // data member (also instance variable)
    String name;
    public static void main(String args[])
    {
        // creating an object of
        // Student
        Student s1 = new Student();
        System.out.println(s1.id);
        System.out.println(s1.name);
    }
}

```

Components of Java Classes

In general, class declarations can include these components, in order:

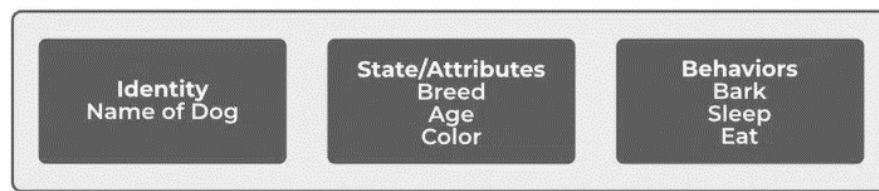
1. **Modifiers:** A class can be public or has default access (Refer this for details).
2. **Class keyword:** class keyword is used to create a class.
3. **Class name:** The name should begin with an initial letter (capitalized by convention).
4. **Superclass (if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. **Interfaces (if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. **Body:** The class body is surrounded by braces, {}.

Java Objects

An object in Java is a basic unit of Object-Oriented Programming and represents real-life entities. Objects are the instances of a class that are created to use the attributes and methods of a class. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

1. **State:** It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior:** It is represented by the methods of an object. It also reflects the response of an object with other objects.
3. **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog

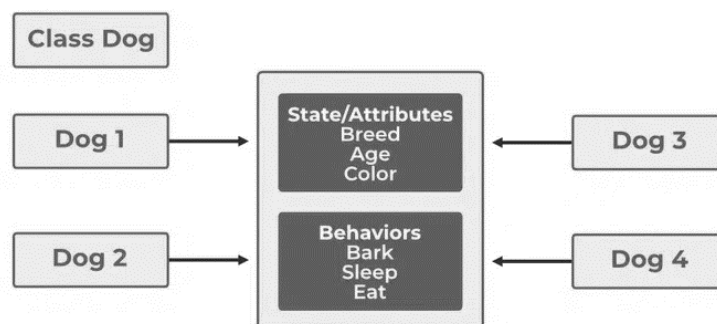


Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, and “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

Creating Objects

When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

Example:



As we declare variables like (type name;). This notifies the compiler that we will use the name to refer to data whose type is type. With a primitive variable, this declaration also reserves the proper amount of memory for the variable. So for reference variables , the type must be strictly a concrete class name. In general, we **can't** create objects of an abstract class or an interface.

Syntax: Dog tuffy;

If we declare a reference variable(tuffy) like this, its value will be undetermined(null) until an object is actually created and assigned to it. Simply declaring a reference variable does not create an object.

Initializing a Java object

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor.

Example:

```
// Class Declaration
```

```

public class Dog
{
    // Instance Variables
    String name;
    String breed;
    int age;
    String color;
    // Constructor Declaration of Class
    public Dog (String name, String breed, int age, String color)
    {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }
    // method 1
    public String getName() { return name; }
    // method 2
    public String getBreed() { return breed; }
    // method 3
    public int getAge() { return age; }
    // method 4
    public String getColor() { return color; }
    @Override public String toString()
    {
        return ("Hi my name is " + this.getName()
            + ".\nMy breed,age and color are "
            + this.getBreed() + "," + this.getAge()
            + "," + this.getColor());
    }
    public static void main(String[] args)
    {
        Dog tuffy
            = new Dog("tuffy", "papillon", 5, "white");
        System.out.println(tuffy.toString());
    }
}

```

Output:

Hi my name is tuffy.

My breed,age and color are papillon,5,white

Ways to Create an Object of a Class

There are four ways to create objects in Java. Strictly speaking, there is only one way (by using a *new* keyword), and the rest internally use a *new* keyword.

i. Using new keyword

It is the most common and general way to create an object in Java.

Example:

```

// creating object of class Test
Test t = new Test();

```

ii. Using Class.forName (String className) method

There is a pre-defined class in java.lang package with name Class. The forName(String className) method returns the Class object associated with the class with the given string name. We have to give a fully qualified name for a class. On calling the new Instance() method on this Class object returns a new instance of the class with the given string name.

```
// creating object of public class Test
// consider class Test present in com.p1 package
Test obj = (Test)Class.forName("com.p1.Test").newInstance();
```

iii. Using clone() method

clone() method is present in the Object class. It creates and returns a copy of the object.

```
// creating object of class Test
Test t1 = new Test();
// creating clone of above object
Test t2 = (Test)t1.clone();
```

Access Specifier in Java

In Java, Access modifiers help to restrict the scope of a class, constructor, variable, method, or data member. It provides security, accessibility, etc to the user depending upon the access modifier used with the element. Let us learn about Java Access Modifiers, their types, and the uses of access modifiers in this article.

Types of Access Modifiers in Java

There are four types of access modifiers available in Java:

1. Default – No keyword required
2. Private
3. Protected
4. Public

1. Default Access Modifier

When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default. The data members, classes, or methods that are not declared using any access modifiers i.e. having default access modifiers are accessible only within the same package. In this example, we will create two packages and the classes in the packages will be having the default access modifiers and we will try to access a class from one package from a class of the second package.

```
// Java program to illustrate default modifier
package p1;
// Class Geek is having Default access modifier
```

```

class Geek
{
    void display()
    {
        System.out.println("Hello World!");
    }
}

```

2. Private Access Modifier

The private access modifier is specified using the keyword private. The methods or data members declared as private are accessible only within the class in which they are declared. Any other class of the same package will not be able to access these members. Top-level classes or interfaces cannot be declared as private because private means “only visible within the enclosing class”. protected means “only visible within the enclosing class and any subclasses”, Hence these modifiers in terms of application to classes, apply only to nested classes and not on top-level classes. In this example, we will create two classes A and B within the same package p1. We will declare a method in class A as private and try to access this method from class B and see the result.

```

// Java program to illustrate error while
// Using class from different package with
// Private Modifier
package p1;
// Class A
class A
{
    private void display()
    {
        System.out.println("GeeksforGeeks");
    }
}
// Class B
class B
{
    public static void main(String args[])
    {
        A obj = new A();
        // Trying to access private method of another class
        obj.display();
    }
}

```

Output:

```

error: display() has private access in A
      obj.display();

```

3. Protected Access Modifier

The protected access modifier is specified using the keyword `protected`. The methods or data members declared as `protected` are accessible within the same package or subclasses in different packages. In this example, we will create two packages `p1` and `p2`. Class `A` in `p1` is made public, to access it in `p2`. The method `display` in class `A` is `protected` and class `B` is inherited from class `A` and this `protected` method is then accessed by creating an object of class `B`.

Example:

```
// Java Program to Illustrate
// Protected Modifier
package p1;
// Class A
public class A
{
    protected void display()
    {
        System.out.println("GeeksforGeeks");
    }
}
```

4 Public Access modifier:

The public access modifier is specified using the keyword `public`. The public access modifier has the widest scope among all other access modifiers. Classes, methods, or data members that are declared as `public` are accessible from everywhere in the program. There is no restriction on the scope of public data members.

Example:

```
// Java program to illustrate
// public modifier
package p1;
public class A
{
    public void display()
    {
        System.out.println("GeeksforGeeks");
    }
}
```

Constructor and Destructor in Java

In Java, a constructor is a particular method that initializes an object when it is first formed. It guarantees that the item begins its trip with predetermined values and configurations. Consider it a blueprint for the object, outlining how it should be initialized. In Java, constructors have the same name as the class they belong to. They have no return type, not

even void, which distinguishes them from conventional methods. When you build an instance of a class using the 'new' keyword, the constructor is automatically invoked, setting the stage for the object to perform its duties.

Syntax:

```
public class Car
{
    // Constructor
    public Car()
    {
        // Initialization logic goes here
    }
}
```

Constructors exist in a variety of types; parameterized constructors enable you to supply data during object formation, whilst default constructors are used when there is no explicit constructor declared. Unlike several programming languages, Java does not have explicit destructors. Instead, it depends on the garbage collector to automatically reclaim memory used by things that are no longer in use. This technique, referred to as garbage collection, relieves the programmer of manual memory management duties.

Java doesn't offer a traditional destructor, developers can implement the `finalize()` method. However, it's essential to note that relying solely on `finalize()` for cleanup is not recommended due to its unpredictable nature.

```
public class Car
{
    // Finalize method for cleanup
    @Override
    protected void finalize() throws Throwable {
        // Cleanup logic goes here
        super.finalize();
    }
}
```

To summarise, constructors give life to Java objects by purposefully initialising them, whereas destructors, or their substitutes, provide a graceful farewell when an object's trip is over. The ability to generate and deconstruct objects is critical for developing strong and efficient Java programs.

Static members in Java

The static keyword in Java is mainly used for memory management. The static keyword in Java is used to share the same variable or method of a given class. The users can apply static keywords with variables, methods, blocks, and nested classes. The static keyword belongs to

the class than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.

The static keyword is a non-access modifier in Java that is applicable for the following:

- Blocks
- Variables
- Methods
- Classes

Note: To create a static member (block, variable, method, nested class), precede its declaration with the keyword static.

Characteristics of static keyword:

Shared memory allocation: Static variables and methods are allocated memory space only once during the execution of the program. This memory space is shared among all instances of the class, which makes static members useful for maintaining global state or shared functionality.

Accessible without object instantiation: Static members can be accessed without the need to create an instance of the class. This makes them useful for providing utility functions and constants that can be used across the entire program.

Associated with class, not objects: Static members are associated with the class, not with individual objects. This means that changes to a static member are reflected in all instances of the class, and that you can access static members using the class name rather than an object reference. Cannot access non-static members: Static methods and variables cannot access non-static members of a class, as they are not associated with any particular instance of the class. Can be overloaded, but not overridden: Static methods can be overloaded, which means that you can define multiple methods with the same name but different parameters. However, they cannot be overridden, as they are associated with the class rather than with a particular instance of the class.

When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object. For example, in the below java program, we are accessing static method m1() without creating any object of the Test class.

```
// Java program to demonstrate that a static member
// can be accessed before instantiating a class
class Test
{
    // static method
    static void m1()
    {
        System.out.println("from m1");
    }
}
```



```
    }  
    public static void main(String[] args)  
    {  
        // calling m1 without creating  
        // any object of class Test  
        m1();  
    }  
}
```

Output:
from m1

Module II

Inheritance

Java, Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in Java by which one class is allowed to inherit the features (fields and methods) of another class. In Java, Inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.

Why Do We Need Java Inheritance?

- **Code Reusability:** The code written in the Superclass is common to all subclasses. Child classes can directly use the parent class code.
- **Method Overriding:** Method Overriding is achievable only through Inheritance. It is one of the ways by which Java achieves Run Time Polymorphism.
- **Abstraction:** The concept of abstract where we do not have to provide all details is achieved through inheritance. Abstraction only shows the functionality to the user.

Base and Derive Class in Java Inheritance

- **Class:** Class is a set of objects which shares common characteristics/ behavior and common properties/ attributes. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
- **Super Class/Parent Class:** The class whose features are inherited is known as a superclass (or a base class or a parent class).
- **Sub Class/Child Class:** The class that inherits the other class is known as a subclass (or a derived class, extended class, or child class). The

subclass can add its own fields and methods in addition to the superclass fields and methods.

- **Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

How to Use Inheritance in Java?

The **extends keyword** is used for inheritance in Java. Using the extends keyword indicates you are derived from an existing class. In other words, “extends” refers to increased functionality.

Syntax:

```
class DerivedClass extends BaseClass
{
    //methods and fields
}
```

Inheritance is one of the key features of OOP that allows us to create a new class from an existing class. The new class that is created is known as subclass (child or derived class) and the existing class from where the child class is derived is known as superclass (parent or base class). The extends keyword is used to perform inheritance in Java.

For example:

```
class Animal
{
    // methods and fields
}

// use of extends keyword
// to perform inheritance
```

```

class Dog extends Animal
{
    // methods and fields of Animal
    // methods and fields of Dog
}

```

In the above example, the Dog class is created by inheriting the methods and fields from the Animal class. Here, Dog is the subclass and Animal is the superclass.

Example 1: Java Inheritance

```

class Animal
{
    // field and method of the parent class
    String name;
    public void eat()
    {
        System.out.println("I can eat");
    }
}

// inherit from Animal
class Dog extends Animal
{
    // new method in subclass
    public void display()
    {
        System.out.println("My name is " + name);
    }
}

class Main
{
    public static void main(String[] args)
    {
        // create an object of the subclass
        Dog labrador = new Dog();
        // access field of superclass
        labrador.name = "Rohu";
        labrador.display();
    }
}

```

```
        // call method of superclass
        // using object of subclass
        labrador.eat();
    }
}
```

Output:

```
My name is Rohu
I can eat
```

Types of Inheritance in Java

Have you ever wondered about the ways in which Java allows for code reusability and hierarchical organization through inheritance? Java supports several types of inheritance: single inheritance through class extension, multilevel inheritance to create a chain of class relationships, hierarchical inheritance for multiple classes to share a single superclass, and multiple inheritance through interfaces for a class to adopt methods from multiple sources. Hybrid inheritance combines these approaches, offering flexibility in complex software design. The following sections elaborate the types of reusability support by java.

- Single-level inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Multiple Inheritance
- Hybrid Inheritance

NOTE: Java does not support Multiple Inheritance and Hybrid Inheritance directly through classes due to its language design, it facilitates aspects of these inheritance types using interfaces and other mechanisms.

Single Inheritance

In this type of java inheritance, the class inherits the properties of some other class. It allows derived classes to take properties and behavior from a single-parent class. In turn, this will make it possible to reuse current code and give it new functionalities. Here, Class A serves as the parent class, while Class B, the child class, inherits the traits and characteristics of the

parent class. The following code shows a comparable idea:

```
class Animal
{
    void eat()
    {
        System.out.println("eating");
    }
}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking");
    }
}
class TestInheritance
{
    public static void main(String args[])
    {
        Dog d=new Dog();
        d.bark();
        d.eat();
    }
}
```

Multi-Level Inheritance

Multi-level type of java inheritance comes with a chain of inheritance. This indicates that we feature a parent class which a derived class inherits. The derived class then serves as the parent to the next class, and so forth. There is a dog class descended from the Animal class, sticking with the Animal class example from below. Another great option is the puppy class – a young dog descended from the Dog class. This way, you can possess a tiered inheritance.

```
class Animal
{
    void eat()
    {
        System.out.println("eating...");
    }
}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking...");
    }
}
class Puppy extends Dog
{
    void weep()
    {
        System.out.println("weeping...");
    }
}
class TestInheritance2
{
    public static void main(String args[])
    {
        Puppy d=new Puppy();
        d.weep();
        d.bark();
        d.eat();
    }
}
```

Multiple Inheritances

The concept of inheritance, which enables classes to adopt features and attributes from other classes, is fundamental to object-oriented programming. Due to Java's support for single inheritance, a class can only descend from one superclass. However, Java offers a method for achieving multiple inheritances through interfaces, enabling a class to implement many interfaces. We will examine the idea of multiple inheritance in Java, how it is implemented using interfaces, and use examples to help us understand.

Understanding Multiple Inheritance A class's capacity to inherit traits from several classes is referred to as multiple inheritances. This notion may be quite helpful when a class needs features from many sources. Multiple inheritances, however, can result in issues like the diamond problem, which occurs when two superclasses share the same method or field and causes conflicts. Java uses interfaces to implement multiple inheritances in order to prevent these conflicts.

Java interfaces

A Java interface is a group of abstract methods that specify the behavior that implementing classes must follow. It serves as a class blueprint by outlining each class's methods. Interfaces offer a degree of abstraction for specifying behaviors but cannot be instantiated like classes. In Java, a class can successfully implement several interfaces to achieve multiple inheritance.

Syntax of implementing multiple interfaces:

```
class MyClass implements Interface1, Interface2, Interface3
{
    // class body
}
```

The classes "MyClass" and "Interface1", "Interface2", and "Interface3" can now inherit and implement methods from other interfaces. As a result, the class is able to display the behaviours specified in every interface it implements.

Example - 1

Let's look at an example situation to demonstrate multiple inheritance using Java interfaces. Imagine that you and I are creating a game with a variety of characters, such as warriors and magicians. We also carry a variety of weaponry, including swords and wands. Although we want to keep character kinds and weapon types apart, we also want our characters to be able to utilize weapons. The following is how multiple inheritances through interfaces may help us

do this:

```
interface Character
```

```
{  
    void attack();  
}
```

```
interface Weapon
```

```
{  
    void use();  
}
```

```
class Warrior implements Character, Weapon
```

```
{  
    public void attack()  
    {  
        System.out.println("Warrior attacks with a sword.");  
    }  
    public void use()  
    {  
        System.out.println("Warrior uses a sword.");  
    }  
}
```

```
class Mage implements Character, Weapon
```

```
{  
    public void attack()  
    {  
        System.out.println("Mage attacks with a wand.");  
    }  
    public void use()  
    {  
        System.out.println("Mage uses a wand.");  
    }  
}
```

```
public class MultipleInheritance
```

```
{
```

```

public static void main(String[] args)
{
    Warrior warrior = new Warrior();
    Mage mage = new Mage();
    warrior.attack(); // Output: Warrior attacks with a sword.
    warrior.use(); // Output: Warrior uses a sword.
    mage.attack(); // Output: Mage attacks with a wand.
    mage.use(); // Output: Mage uses a wand.
}
}

```

Output:

```

Warrior attacks with a sword.
Warrior uses a sword.
Mage attacks with a wand.
Mage uses a wand.

```

Explanation: The interfaces "Character" and "Weapon" in the example above specify the behaviour that classes that implement them must have. As a result of the classes "Warrior" and "Mage" implementing both interfaces, the necessary behaviors may be inherited and shown. The main method shows how to instantiate these classes' objects and call their corresponding behaviors.

Hierarchical Inheritance

This type of java inheritance is where many subclasses inherit from one single class. Basically it is a combination of more than one type of java inheritance. When a class contains several child classes or subclasses, or, to put it another way, when multiple child classes share the same parent class, this type of inheritance is referred to as hierarchical.

class Animal

```

{
    void eat()
    {
        System.out.println("eating...");
    }
}

```

```

}
class Dog extends Animal
{
    void bark()
    {
        System.out.println("barking...");
    }
}
class Cat extends Animal
{
    void meow()
    {
        System.out.println("meowing...");
    }
}
class TestInheritance3
{
    public static void main(String args[])
    {
        Cat c=new Cat();
        c.meow();
        c.eat();
    }
}

```

Hybrid Inheritance

Hybrid type of java inheritance is a combination of more than two types of java inheritances single and multiple. A hybrid inheritance combines a single or more of the inheritance types we've covered so far. Any combination, though, leads to a form of multiple inheritances that Java does not support.

Hybrid Inheritance Example:

Class A and B extends class C → Hierarchical inheritance

Class D extends class A → Single inheritance

```
class C
```

```
{  
    public void disp()  
    {  
        System.out.println("C");  
    }  
}
```

```
class A extends C
```

```
{  
    public void disp()  
    {  
        System.out.println("A");  
    }  
}
```

```
class B extends C
```

```
{  
    public void disp()  
    {  
        System.out.println("B");  
    }  
}
```

```
class D extends A
```

```
{  
    public void disp()  
    {  
        System.out.println("D");  
    }  
    public static void main(String args[]){
```

```
D obj = new D();
obj.disp();
}
}
```

Output: D

Access control in inheritance

Java provides a rich set of modifiers. They are used to control access mechanisms and also provide information about class functionalities to JVM. They are divided into two categories namely:

- **Access modifiers**
- **Non-access modifiers**

Java's access modifiers are public, private, and protected. Java also defines a default access level (called package-private).

- **public:** When a member of a class is modified by public, then that member can be accessed by any other code.
- **private:** When a member of a class is specified as private, then that member can only be accessed by other members of its class.
- **default:** It is also referred to as no modifier. Whenever we do not use any access modifier it is treated as default where this allows us to access within a class, within a subclass, and also non-sub class within a package but when the package differs now be it a subclass or non-class we are not able to access.
- **protected:** With the above default keyword we were facing an issue as we are getting closer to the real world with the above default modifier but there was a constriction as we are not able to access class sub-class from a different package. So protected access modifier allows not only to access class be it subclass or non-sub class but allows us to access subclass of the different package which brings us very close to a real-world and hence strong understanding of inheritance is required for understanding and implementing this keyword.

The following table elaborate the details about access modifiers provided.

	default	private	protected	public
same class	yes	yes	yes	yes
same package subclass	yes	no	yes	yes
same package non-subclass	yes	no	yes	yes
different package subclass	no	no	yes	yes
different package non-subclass	no	no	no	yes

Note: Now you can understand why `main( )` has always been preceded by the `public` modifier. It is called by code that is outside the program—that is, by the Java run-time system. When no access modifier is used, then by default the member of a class is `public` within its own package, but cannot be accessed outside of its package. `protected` applies only when inheritance is involved.

Polymorphism

Polymorphism is derived from two Greek words, “poly” and “morph”, which mean “many” and “forms”, respectively. Hence, polymorphism meaning in Java refers to the ability of objects to take on many forms. In other words, it allows different objects to respond to the same message or method call in multiple ways.

Polymorphism in Java Example

As previously explained, polymorphism in Java helps an object take on many different forms. In this section, we will provide different examples of polymorphism to show how it works. The `Animal` class has a `makeSound()` method that outputs “Animal making a sound...” while the subclasses `Dog`, `Cat`, and `Elephant`, each provide their own implementation of the same function to produce individual noises.

```
class Animal
{
    void makeSound()
    {
        System.out.println("Animal making a sound...");
    }
}

class Dog extends Animal
{

```

```

void makeSound() {
    System.out.println("Dog barking...");
}
}
class Cat extends Animal
{
    void makeSound()
    {
        System.out.println("Cat meowing...");
    }
}
class Elephant extends Animal
{
    void makeSound()
    {
        System.out.println("Elephant trumpeting...");
    }
}
class TestPolymorphism2
{
    public static void main(String args[])
    {
        Animal animal;
        animal = new Dog();
        animal.makeSound();
        animal = new Cat();
        animal.makeSound();
        animal = new Elephant();
        animal.makeSound();
    }
}

```

Output:

Dog barking...

Cat meowing...

Elephant trumpeting...

Function Overloading in Java

Function Overloading in Java occurs when there are functions having the same name but have different numbers of parameters passed to it, which can be different in data like int, double, float and used to return different values are computed inside the respective overloaded method. Function overloading is used to reduce complexity and increase the efficiency of the program by involving more functions that are segregated and can be used to distinguish among each other with respect to their individual functionality. Overloaded functions are related to compile-time or static polymorphism. There is also a concept of type conversion, which is basically used in overloaded functions used to calculate the conversion of type in variables.

Overloaded functions have the same name but different types of arguments or parameters assigned to them. They can be used to calculate mathematical or logical operations within the number of assigned variables in the method. The syntax of the overloaded function can be given below, where there are up to N number of variables assigned.

Syntax:

```
public class OverloadedMethod
{
    public int FunctionName(int x, int y) //Two parameters in the function
    {
        return (x + y); //Returns the sum of the two numbers
    }
    // This function takes three integer parameters
    public int FunctionName(int x, int y, int z)
    {
        return (x + y + z);
    }
    // This function takes two double parameters
    public double FunctionName(double x, double y)
    {
        return (x + y);
    }
    //Many more such methods can be done with different number of parameters
    // Code used to input the number and
    public static void main(String args[])
    {
        FunctionName s = new FunctionName();
        System.out.println(s.FunctionName(10, 20));
        System.out.println(s.FunctionName(10, 20, 30));
        System.out.println(s.FunctionName(10.5, 20.5));
    }
}
```


Explanation: Function overloading works by calling different functions having the same name, but the different number of arguments passed to it. There are many coding examples that can be shown in order to identify the benefits and disadvantages of function overloading properly.

Operator overloading in Java

Operator overloading aims to redefine an operator that has been defined and has certain functions to complete more detailed and specific operations and other functions. From an object-oriented perspective, it means an operator can be defined as a method of a class, so the function of the operator can be used to represent a certain behaviour of the object.

There are at least two benefits to being able to perform operator overloading for numeric operations of non-primitive types.

1. The code is simpler to write and less error-prone.
2. The code is easier to read without many parentheses.

How to Implement Operator Overloading in Java

The implementation of operator overloading in Java still uses Manifold. Manifold allows you to overload Java operators in various scenarios, such as arithmetic operators (including +, -, *, /, and %), comparison operators (>, >=, <, <=, ==, and !=), and index operators ([]). Please see Java's Missing Feature: Extension Methods for more information about the integration of Manifold.

Arithmetic Operator

Manifold is a function that maps each overload of an arithmetic operator to a specific name. For example, if you define a plus(B) method in class A, that class can be called using a + b instead of a.plus(b). The following chart describes the mappings:

Operator	Method Call
c = a + b	c = a.plus(b)
c = a – b	c = a.minus(b)
c = a * b	c = a.times(b)
c = a / b	c = a.div(b)
c = a % b	c = a.rem(b)

Those familiar with Kotlin should know that this is an imitation of Kotlin's operator overloading.

Let's define a numeric Num to facilitate illustration.

```
public class Num
{
    private final int v;
    public Num(int v)
    {
        this.v = v;
    }

    public Num plus(Num that)
    {
        return new Num(this.v + that.v);
    }

    public Num minus(Num that)
    {
        return new Num(this.v - that.v);
    }

    public Num times(Num that)
    {
        return new Num(this.v * that.v);
    }
}
```

For the following code:

```
Num a = new Num(1);
Num b = new Num(2);
Num c = a + b - a;
```

Virtual functions and runtime polymorphism

A member function that has the keyword virtual used in its declaration in the base class and is redefined (Overridden) in the derived class is referred to as a virtual function. The late binding instruction instructs the compiler to execute the called function during runtime by matching the object with the appropriately called function. Runtime Polymorphism refers to this method.

1. No matter what kind of reference (or pointer) is used to invoke a function, virtual functions make sure the right function is called for an object.
2. Their primary purpose is to implement runtime polymorphism.
3. In base classes, functions are declared using the virtual keyword.
4. Runtime resolution of function calls is carried out.

Polymorphism is a term used to describe the capacity to assume several shapes. If there is a hierarchy of classes connected to one another by inheritance, it happens. Polymorphism, which is defined as "showing diverse traits in different contexts," can be summarised as "showing different characteristics in a variety of situations" and "polymorphism."

What is the use of virtual functions?

To achieve Runtime Polymorphism, virtual functions are primarily used. Only a base class type pointer (or reference) can enable runtime polymorphism. A base class pointer can also point to both objects from the base class and those from derived classes.

Also, without even knowing the type of derived class object, we can use virtual functions to compile a list of base class pointers and call any of the derived classes' methods.

```
#include<iostream>
using namespace std;
class B
{
public:
    virtual void s()
    {
        cout<<" In Base \n";
    }
};
class D: public B
{
public:
    void s()
    {
        cout<<"In Derived \n";
    }
};
int main(void)
{
    D d; // An object of class D
    B *b= &d; // A pointer of type B* pointing to d
    b->s(); // prints "D::s() called"
    return 0;
}
```

Output: In Derived

What are the rules for virtual functions?

- I. Virtual functions are not permitted to be static or friendly to other classes.
- II. Pointers or references of base class type are required to access virtual functions.
- III. Both the base class and any derived classes should use the same function prototype.
- IV. There cannot be a virtual constructor in a class. However, it might have a virtual

destroyer.

- v. The base class always defines them, and the derived class redefines them.

What is runtime polymorphism?

Runtime polymorphism is the process of binding an object at runtime with a capability. Overriding methods is one way to implement runtime polymorphism. At runtime, not at compilation time, the Java virtual machine decides which method to invoke. Additionally known as dynamic binding or late binding. The parent class's method is overridden in the child class, according to this concept. The term "method overriding" refers to the situation where a child class implements a method specifically that was supplied by one of its parent classes. You can see runtime polymorphism in the example that follows.

Example

```
class Test
{
    public void method()
    {
        System.out.println("Method 1");
    }
}
public class DEMO extends Test
{
    public void method()
    {
        System.out.println("Method 2");
    }
    public static void main(String args[])
    {
        Test test = new DEMO();
        test.method();
    }
}
```

Output: Method 2

What are the limitations of virtual functions?

Slower: The virtual mechanism causes the function call to take a little longer, making it harder for the compiler to optimize as it is unsure which function will be called at compilation time. Virtual functions can make it slightly more challenging to determine where a function is being called from in complicated systems, which makes them more challenging to debug.

Abstract Class in Java

In Java, abstract class is declared with the abstract keyword. It may have both abstract and non-abstract methods (methods with bodies). An abstract is a Java modifier applicable for classes and methods in Java but not for Variables. In this article, we will learn the use of abstract classes in Java. Furthermore, Java abstract class is a class that cannot be initiated by itself, it needs to be subclassed by another class to use its properties. An abstract class is declared using the “abstract” keyword in its class definition.

Illustration of Abstract class

```
abstract class Shape
{
    int color;
    // An abstract function
    abstract void draw();
}
```

In Java, the following some important observations about abstract classes are as follows:

1. An instance of an abstract class can not be created.
2. Constructors are allowed.
3. We can have an abstract class without any abstract method.
4. There can be a final method in abstract class but any abstract method in class (abstract class) can not be declared as final or in simpler terms final method can not be abstract itself as it will yield an error: “Illegal combination of modifiers: abstract and final”.
5. We can define static methods in an abstract class.
6. We can use the abstract keyword for declaring top-level classes (Outer class) as well as inner classes as abstract.
7. If a class contains at least one abstract method then compulsory should declare a class as abstract.
8. If the Child class is unable to provide implementation to all abstract methods of the Parent class then we should declare that Child class as abstract so that the next level Child class should provide implementation to the remaining abstract method.

Example of Java Abstract Class

```
// Abstract class
abstract class Sunstar
{
    abstract void printInfo();
}

// Abstraction performed using extends
class Employee extends Sunstar {
    void printInfo()
    {
        String name = "avinash";
        int age = 21;
        float salary = 222.2F;

        System.out.println(name);
        System.out.println(age);
        System.out.println(salary);
    }
}

// Base class
class Base {
    public static void main(String args[])
    {
        Sunstar s = new Employee();
        s.printInfo();
    }
}
```

Output:

```
avinash
21
222.2
```

Pure Virtual Function

Pure virtual function is a virtual function for which we don't have implementations. An abstract method in Java can be considered as a pure virtual function. Let's take an example to understand this better.

Example of Pure Virtual Function:

```
abstract class Dog
{
```

```

        final void bark()
        {
            System.out.println("woof");
        }

        abstract void jump(); //this is a pure virtual function
    }
class MyDog extends Dog
{
    void jump()
    {
        System.out.println("Jumps in the air");
    }
}
public class Runner
{
    public static void main(String args[])
    {
        Dog ob1 = new MyDog();
        ob1.jump();
    }
}

```

Output: Jumps in the air

This is how virtual function can be used with abstract class.

Run-Time Polymorphism

Run-time polymorphism is when a call to an overridden method is resolved at run-time instead of [compile-time](#). The overridden method is called through the reference variable of the base class.

Output: Java Certification Course

```
class Edureka
{
    public void show()
    {
        System.out.println("welcome to edureka");
    }
}
class Course extends Edureka
{
    public void show()
    {
        System.out.println("Java Certification Program");
    }
}
public static void main(String args[])
{
    Edureka ob1 = new Course();
    ob1.show();
}
}
```

Points To Remember

- For a virtual function in Java, you do not need an explicit declaration. It is any function that we have in a base class and redefined in the derived class with the same name.
- The base class pointer can be used to refer to the object of the derived class.
- During the execution of the program, the base class pointer is used to call the derived class functions.

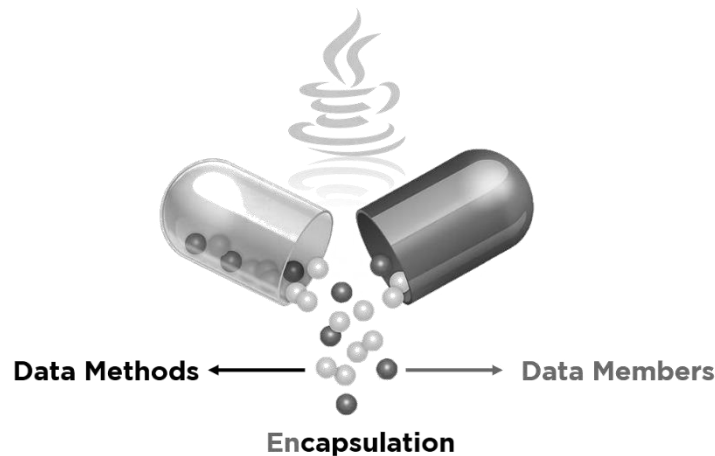
This brings us to the end of this article where we have learned about the Virtual Function In Java. I hope you are clear with all that has been shared with you in this tutorial.

Section B

Module III

Encapsulation

Encapsulation is a powerful mechanism for storing the data members and data methods of a class together. It is done in the form of a secure field accessible by only the members of the same class. Encapsulation in Java refers to integrating data (variables) and code (methods) into a single unit. In encapsulation, a class's variables are hidden from other classes and can only be accessed by the methods of the class in which they are found.



Source: simplilearn

Encapsulation in Java is an object-oriented procedure of combining the data members and data methods of the class inside the user-defined class. It is important to declare this class as private. It refers to the bundling of data and methods that operate on the data within a single unit, typically a class. This concept helps in hiding the internal state of an object and only exposing necessary functionalities through methods. By encapsulating data, Java ensures better data security and code maintainability. Understanding encapsulation is fundamental for building robust and organized Java applications, making a Java Course essential for mastering this concept. Next, we will understand the Syntax to be followed while implementing encapsulation in Java.

Syntax:

```
<Access_Modifier> class <Class_Name>
{
    private <Data_Members>;
    private <Data_Methods>;
}
```

For enhancing the understanding of the encapsulation process, let us go through the following sample program.

Example:

```
package dc;
public class c
{
    public static void main (String[] args)
    {
        Employee e = new Employee();
        e.setName("Robert");
        e.setAge(33);
        e.setEmpID(1253);
        System.out.println("Employee's name: " + e.getName());
        System.out.println("Employee's age: " + e.getAge());
        System.out.println("Employee's ID: " + e.getEmpID());
    }
}

package dc;

public class Employee
{
    private String Name;
    private int EmpID;
    private int Age;
    public int getAge()
```

```
{  
    return Age;  
}  
public String getName()  
{  
    return Name;  
}  
public int getEmpID()  
{  
    return EmpID;  
}  
public void setAge(int newAge)  
{  
    Age = newAge;  
}  
public void setName(String newName)  
{  
    Name = newName;  
}  
public void setRoll(int newEmpID)  
{  
    EmpID = newEmpID;  
}  
public void setEmpID(int EmpID)  
{  
}  
}
```

Output:

Employee's name: Robert

Employee's age: 33

Employee's ID: 1253

Data Hiding in Java

Data hiding is a procedure done to avoid access to the data members and data methods and their logical implementation. Data hiding can be done by using the access specifiers. We have four access specifiers, which are as follows.



Default

Default is the first line of data hiding. If any class in Java is not mentioned with an access specifier, then the compiler will set 'default' as the access specifier. The access specifications of default are extremely similar to that of the public access specifier.

Public

The public access specifier provides the access specifications to a class so that it can be accessed from anywhere within the program.

Example:

```
package Simplilearn;
class vehicle
{
    public int tires;
    public void display()
    {
        System.out.println("I have a vehicle.");
        System.out.println("It has " + tires + " tires.");
    }
}
```

```
public class Display
{
    public static void main(String[] args)
    {
        vehicle veh = new vehicle();
        veh.tires = 4;
        veh.display();
    }
}
```

//Output:

I have a vehicle.

It has four tires.

Private

The private access specifier provides access to the data members, and the data methods limit to the class itself.

Example:

```
package Simplilearn;
class Student
{
    private int rank;
    public int getRank()
    {
        return rank;
    }
    public void setRank(int rank)
    {
        this.rank = rank;
    }
}
```

```

    }
}
public class school
{
    public static void main(String[] args)
    {
        Student s = new Student();
        s.setRank(1022);
        System.out.println("Student rank is " + s.getRank());
    }
}

```

//Output:

Student rank is 1022

Protected

The protected access specifier protects the class methods and members similar to the private access specifier. The main difference is that the access is limited to the entire package, unlike only a class with the private access specifier.

Example:

```

package Simplilearn;
class human
{
    protected String stream;
    protected void display()
    {
        System.out.println("Hello, I am a " + stream + " Student");
    }
}

public class Student extends human
{

```

```
public static void main(String[] args)
{
    Student s = new Student();
    s.stream = "Computer Science and Engineering Technology";
    s.display();
}
}
```

//Output:

Hello, I am a Computer Science and Engineering Technology Student

Accessor and Mutator methods in Java

In object-oriented programming, encapsulation is a fundamental concept that refers to the practice of hiding the implementation details of an object and providing an interface to access its properties and behaviors. Accessor and mutator methods are two important concepts related to encapsulation in Java.

Accessor

Accessor methods, also known as getter methods, are methods that allow you to retrieve the value of an object's private instance variables. These methods provide read-only access to the object's state. By using accessor methods, you can ensure that the object's state is not modified accidentally or maliciously by external code.

Mutator

Mutator methods, also known as setter methods, are methods that allow you to modify the value of an object's private instance variables. These methods provide write-only access to the object's state. By using mutator methods, you can ensure that the object's state is modified only through a controlled interface.

Let's take a look at an example to understand the concept of accessor and mutator methods in

Java. Suppose we have a class called Person that has three private instance variables: name, age, and email. We want to provide access to these variables using accessor and mutator methods.

```
public class Person
{
    private String name;
    private int age;
    private String email;

    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public int getAge()
    {
        return age;
    }
    public void setAge(int age)
    {
        this.age = age;
    }
    public String getEmail()
    {
        return email;
    }
    public void setEmail(String email)
    {
        this.email = email;
    }
}
```



```
}
```

In this example, we have defined three accessor methods: `getName()`, `getAge()`, and `getEmail()`, and three mutator methods: `setName()`, `setAge()`, and `setEmail()`. The accessor methods return the value of the corresponding instance variable, while the mutator methods set the value of the corresponding instance variable.

Naming Convention

The naming convention for accessor and mutator methods is important in Java. Accessor methods should be named starting with "get" followed by the name of the variable, with the first letter capitalized. Mutator methods should be named starting with "set" followed by the name of the variable, with the first letter capitalized. This naming convention makes it clear what each method does and makes the code more readable and maintainable.

Accessor and mutator methods are an essential part of encapsulation in Java. They allow you to control access to an object's state, ensuring that it is accessed and modified only through a controlled interface. By following the naming convention for accessor and mutator methods, we can make your code more readable and maintainable.

Example:

Person.java

```
import java.util.Scanner;

public class Person
{
    private String name;
    private int age;
    private String email;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
```

```
{
    this.name = name;
}
public int getAge()
{
    return age;
}
public void setAge(int age)
{
    this.age = age;
}
public String getEmail()
{
    return email;
}
public void setEmail(String email)
{
    this.email = email;
}
public static void main(String[] args)
{
    Scanner scanner = new Scanner(System.in);
    Person person = new Person();
    System.out.print("Enter name: ");
    String name = scanner.nextLine();
    person.setName(name);
    System.out.print("Enter age: ");
    int age = scanner.nextInt();
    person.setAge(age);
    scanner.nextLine(); // Consume the newline character left by nextInt()
    System.out.print("Enter email: ");
    String email = scanner.nextLine();
    person.setEmail(email);
    System.out.println("\nName: " + person.getName());
}
```

```
        System.out.println("Age: " + person.getAge());
        System.out.println("Email: " + person.getEmail());
    }
}
```

Output:

Name: Manoj

Age: 21

Email: manoj@gmail.com

In this example, the main method creates a new instance of the Person class and uses the mutator methods (setName, setAge, and setEmail) to set the values of the object's private instance variables. Then, the accessor methods (getName, getAge, and getEmail) are used to retrieve the values of the instance variables and display them on the console.

Advantage of Using Accessor and Mutator

Advantage of using accessor and mutator methods is that they allows us to add validation and other processing logic to the process of getting and setting an object's state. For example, you could add a validation check to the "setAge(int age)" method to ensure that the age value is within a certain range, or add formatting logic to the "getName()" method to capitalize the first letter of the person's name.

Benefits of Encapsulation

Implementing the process of encapsulation in Java has proven to be highly effective and beneficial while programming in real-time. The following are the significant benefits of encapsulation.

- A class can have complete control over its data members and data methods.
- The class will maintain its data members and methods as read-only.
- Data hiding prevents the user from the complex implementations in the code.
- The variables of the class can be read-only or write-only as per the programmer's requirement.
- Encapsulation in Java provides an option of code-reusability.
- Using encapsulation will help in making changes to an existing code quickly.

- Unit testing a code designed using encapsulation is elementary.
- Standard IDEs have the support of getters and setters; this makes coding even faster.

Interfaces and Abstract Classes

Interfaces and abstraction are essential features of object-oriented programming. They provide a way to define contracts and hide implementation details. In Java, they play a crucial role in achieving code flexibility and maintainability.

In Java, interfaces and abstraction are powerful concepts that enable developers to design and implement flexible and extensible software. In this comprehensive guide, we'll delve into the concepts of interfaces and abstraction, provide code examples, discuss key differences, explore new features in Java, and offer best practices to make your Java code more robust and maintainable.

Interface: An Introduction

An interface is a contract that specifies a set of methods without providing their implementations. It acts as a blueprint for classes that implement it. Interfaces enable multiple classes to share a common set of methods without forcing them into a specific inheritance hierarchy.

Declaring interfaces:

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
interface <interface_name>
{

    // declare constant fields

    // declare methods that abstract

    // by default.
```

```
}
```

Defining an Interface

In Java, you define an interface using the interface keyword:

```
public interface Shape
{
    double area();
    void draw();
}
```

Implementing an Interface:

A class implements an interface by using the implements keyword. It must provide implementations for all the methods defined in the interface:

```
public class Circle implements Shape
{
    private double radius;
    public Circle(double radius)
    {
        this.radius = radius;
    }
    @Override
    public double area()
    {
        return Math.PI * radius * radius;
    }
    @Override
    public void draw()
    {
        System.out.println("Drawing a circle");
    }
}
```

Extending Interfaces

One interface can inherit another by the use of keyword extends. When a class implements an interface that inherits another interface, it must provide an implementation for all methods required by the interface inheritance chain.

Example 1:

```
interface A
{
    void method1();
    void method2();
}

// B now includes method1 and method2

interface B extends A
{
    void method3();
}

// the class must implement all method of A and B.

class gfg implements B
{
    public void method1()
    {
        System.out.println("Method 1");
    }

    public void method2()
    {
        System.out.println("Method 2");
    }

    public void method3()
```

```
        {  
            System.out.println("Method 3");  
        }  
    }  
}
```

Example 2:

interface Student

```
{  
    public void data();  
}
```

class avi implements Student

```
{  
    public void data ()  
    {  
        String name="avinash";  
        int rollno=68;  
        System.out.println(name);  
        System.out.println(rollno);  
    }  
}
```

public class inter_face

```
{  
    public static void main (String args [])  
    {  
        avi h= new avi();  
        h.data();  
    }  
}
```

Output:

avinash

68

In a Simple way, the interface contains multiple abstract methods, so write the implementation in implementation classes. If the implementation is unable to provide an implementation of all abstract methods, then declare the implementation class with an abstract modifier, and complete the remaining method implementation in the next created child classes. It is possible to declare multiple child classes but at final we have completed the implementation of all abstract methods.\

In general, the development process is step by step:

- Level 1 – interfaces: It contains the service details.
- Level 2 – abstract classes: It contains partial implementation.
- Level 3 – implementation classes: It contains all implementations.
- Level 4 – Final Code / Main Method: It have access of all interfaces data.

Key Points to Remember (Interface)

- Interfaces define a contract with method signatures but no method bodies.
- A class can implement multiple interfaces.
- Interfaces are used to achieve multiple inheritances in Java.
- Interfaces can be used to define constants (variables with public static final).

Advantages of Interfaces in Java

The advantages of using interfaces in Java are as follows:

- Without bothering about the implementation part, we can achieve the security of

the implementation.

- In Java, multiple inheritances are not allowed, however, you can use an interface to make use of it as you can implement more than one interface.

Abstraction: An Introduction

Abstraction is the process of hiding complex implementation details and showing only the necessary features of an object. In Java, you achieve abstraction using abstract classes and methods.

Abstract Classes

An abstract class is a class that cannot be instantiated but can contain abstract methods (methods without implementation) and concrete methods (methods with implementation).

```
public abstract class Vehicle
{
    public abstract void start();
    public void stop()
    {
        System.out.println("Vehicle stopped");
    }
}
```

Abstract Methods

An abstract method is declared using the abstract keyword and must be implemented by any concrete subclass.

```
public class Car extends Vehicle
{
    @Override
    public void start()
    {
        System.out.println("Car started");
    }
}
```

}

Key Points to Remember:

- Abstract classes can't be instantiated.
- Abstract classes can have both abstract and concrete methods.
- Subclasses of an abstract class must implement its abstract methods.
- Abstraction is used to hide implementation details.

Key Differences

1. Interface vs. Abstract Class

An interface can't have instance variables, while an abstract class can. A class can implement multiple interfaces, but it can extend only one abstract class. Interfaces provide a strong contract, and multiple inheritance is achieved through them.

2. Method Definitions

Interfaces define methods without implementation, while abstract classes can contain both abstract and concrete methods.

3. Usage

Use interfaces when you want to define a contract for a group of classes. Use abstract classes when you want to provide a common base class with some default behavior.

Module IV

Exception Handling

An *exception* is an error event that can happen during the execution of a program and disrupts its normal flow. Java provides a robust and object-oriented way to handle exception scenarios known as Java Exception Handling.

Exceptions in Java can arise from different kinds of situations such as wrong data entered by the user, hardware failure, network connection failure, or a database server that is down. The code that specifies what to do in specific exception scenarios is called exception handling.

Throwing and Catching Exceptions

Java creates an *exception object* when an error occurs while executing a statement. The exception object contains a lot of debugging information such as method hierarchy, line number where the exception occurred, and type of exception.

If an exception occurs in a method, the process of creating the exception object and handing it over to the runtime environment is called “*throwing the exception*”. The normal flow of the program halts and the Java Runtime Environment (JRE) tries to find the handler for the exception. Exception Handler is the block of code that can process the exception object.

- The logic to find the exception handler begins with searching in the method where the error occurred.
- If no appropriate handler is found, then it will move to the caller method.
- And so on.

So, if the method's call stack is A->B->C and an exception is raised in method C, then the search for the appropriate handler will move from C->B->A.

If an appropriate exception handler is found, the exception object is passed to the handler to process it. The handler is said to be “*catching the exception*”. If there is no appropriate exception handler, found then the program terminates and prints information about the exception to the console. Java Exception handling framework is used to handle runtime errors only. The compile-time errors have to be fixed by the developer writing the code else the program won't execute.

Java Exception Handling Keywords

Java provides specific keywords for exception handling purposes.

1. **throw** – We know that if an error occurs, an exception object is getting created and then Java runtime starts processing to handle them. Sometimes we might want to generate exceptions explicitly in our code. For example, in a user authentication program, we should throw exceptions to clients if the password is null. The throw keyword is used to throw exceptions to the runtime to handle it.
2. **throws** – When we are throwing an exception in a method and not handling it, then we have to use the throws keyword in the method signature to let the caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate them to its caller method using

the throws keyword. We can provide multiple exceptions in the throws clause, and it can be used with the main() method also.

3. **try-catch** – We use the try-catch block for exception handling in our code. try is the start of the block and catch is at the end of the try block to handle the exceptions. We can have multiple catch blocks with a try block. The try-catch block can be nested too. The catch block requires a parameter that should be of type Exception.
4. **finally** – the finally block is optional and can be used only with a try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use the finally block. The finally block always gets executed, whether an exception occurred or not.

Example:

ExceptionHandling.java

```
package com.journaldev.exceptions;
import java.io.FileNotFoundException;
import java.io.IOException;
public class ExceptionHandling
{

    public static void main(String[] args) throws FileNotFoundException, IOException
    {
        try
        {
            testException(-5);
            testException(-10);
        }
        catch(FileNotFoundException e)
        {
            e.printStackTrace();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

```

    }
    finally
    {
        System.out.println("Releasing resources");
    }
    testException(15);
}

public static void testException(int i) throws FileNotFoundException, IOException
{
    if (i < 0)
    {
        FileNotFoundException myException = new FileNotFoundException
        ("Negative Integer " + i);
        throw myException;
    }
    else if (i > 10)
    {
        throw new IOException("Only supported for index 0 to 10");
    }
}
}

```

Explanations:

- The testException() method is throwing exceptions using the throw keyword. The method signature uses the throws keyword to let the caller know the type of exceptions it might throw.
- In the main() method, I am handling exceptions using the try-catch block in the main() method. When I am not handling it, I am propagating it to runtime with the throws clause in the main() method.
- The testException(-10) never gets executed because of the exception and then the finally block is executed.

The `printStackTrace()` is one of the useful methods in the `Exception` class for debugging purposes.

This code will output the following:

Output:

```
java.io.FileNotFoundException: Negative Integer -5
    at
com.journaldev.exceptions.ExceptionHandling.testException(ExceptionHandling.java:24)
    at com.journaldev.exceptions.ExceptionHandling.main(ExceptionHandling.java:10)
Releasing resources
Exception in thread "main" java.io.IOException: Only supported for index 0 to 10
    at
com.journaldev.exceptions.ExceptionHandling.testException(ExceptionHandling.java:27)
    at com.journaldev.exceptions.ExceptionHandling.main(ExceptionHandling.java:19)
```

Some important points to note:

- We can't have catch or finally clause without a try statement.
- A try statement should have either catch block or finally block, it can have both blocks.
- We can't write any code between try-catch-finally blocks.
- We can have multiple catch blocks with a single try statement.
- try-catch blocks can be nested similar to if-else statements.
- We can have only one finally block with a try-catch statement.

Java Exception Hierarchy

As stated earlier, when an exception is raised an exception object is getting created. Java Exceptions are hierarchical and inheritance is used to categorize different types of exceptions. `Throwable` is the parent class of Java Exceptions Hierarchy and it has two child objects – `Error` and `Exception`. Exceptions are further divided into `Checked Exceptions` and

Runtime Exceptions.

1. **Errors:** Errors are exceptional scenarios that are out of the scope of application, and it's not possible to anticipate and recover from them. For example, hardware failure, Java virtual machine (JVM) crash, or out-of-memory error. That's why we have a separate hierarchy of Errors and we should not try to handle these situations. Some of the common Errors are `OutOfMemoryError` and `StackOverflowError`.
2. **Checked Exceptions:** Checked Exceptions are exceptional scenarios that we can anticipate in a program and try to recover from it. For example, `FileNotFoundException`. We should catch this exception and provide a useful message to the user and log it properly for debugging purposes. The `Exception` is the parent class of all Checked Exceptions. If we are throwing a Checked Exception, we must catch it in the same method, or we have to propagate it to the caller using the `throws` keyword.
3. **Runtime Exception:** Runtime Exceptions are caused by bad programming. For example, trying to retrieve an element from an array. We should check the length of the array first before trying to retrieve the element otherwise it might throw `ArrayIndexOutOfBoundsException` at runtime. `RuntimeException` is the parent class of all Runtime Exceptions. If we are throwing any Runtime Exception in a method, it's not required to specify them in the method signature `throws` clause. Runtime exceptions can be avoided with better programming.

Custom Exception

In Java, we can create our own exceptions that are derived classes of the `Exception` class. Creating our own `Exception` is known as custom exception or user-defined exception. Basically, Java custom exceptions are used to customize the exception according to user need.

Consider the example in which `InvalidAgeException` class extends the `Exception` class. Using the custom exception, we can have your own exception and message. Here, we have passed a string to the constructor of superclass i.e. `Exception` class that can be obtained using `getMessage()` method on the object we have created.

Why use custom exceptions?

Java exceptions cover almost all the general type of exceptions that may occur in the programming. However, we sometimes need to create custom exceptions.

Following are few of the reasons to use custom exceptions:

To catch and provide specific treatment to a subset of existing Java exceptions. Business logic exceptions: These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem. In order to create custom exception, we need to extend Exception class that belongs to java.lang package.

Consider the following example, where we create a custom exception named WrongFileNameException:

```
public class WrongFileNameException extends Exception
{
    public WrongFileNameException(String errorMessage)
    {
        super(errorMessage);
    }
}
```

Example:

Let's see a simple example of Java custom exception. In the following code, constructor of InvalidAgeException takes a string as an argument. This string is passed to constructor of parent class Exception using the super() method. Also the constructor of Exception class can be called without using a parameter and calling super() method is not mandatory.

TestCustomException1.java

```
// class representing custom exception
class InvalidAgeException extends Exception
{
    public InvalidAgeException (String str)
    {
```



```

        // calling the constructor of parent Exception
        super(str);
    }
}

// class that uses custom exception InvalidAgeException
public class TestCustomException1
{

    // method to check the age
    static void validate (int age) throws InvalidAgeException
    {
        if(age < 18)
        {
            // throw an object of user defined exception
            throw new InvalidAgeException("age is not valid to vote");
        }
        else
        {
            System.out.println("welcome to vote");
        }
    }

}

// main method
public static void main(String args[])
{
    try
    {
        // calling the method
        validate(13);
    }
    catch (InvalidAgeException ex)
    {
        System.out.println("Caught the exception");
    }
}

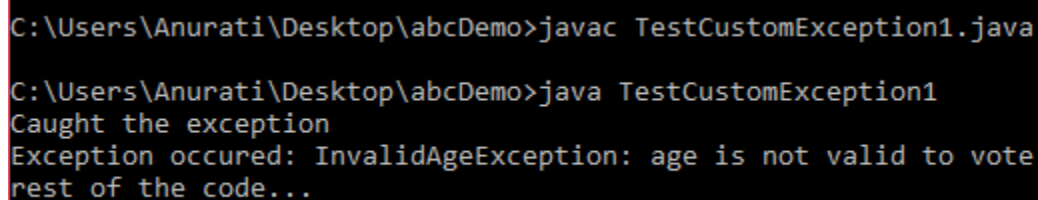
```

```

        // printing the message from InvalidAgeException object
        System.out.println("Exception occurred: " + ex);
    }
    System.out.println("rest of the code...");
}
}

```

Output:



```

C:\Users\Anurati\Desktop\abcDemo>javac TestCustomException1.java
C:\Users\Anurati\Desktop\abcDemo>java TestCustomException1
Caught the exception
Exception occurred: InvalidAgeException: age is not valid to vote
rest of the code...

```

File Handling in Java

In Java, with the help of File Class, we can work with files. This File Class is inside the java.io package. The File class can be used by creating an object of the class and then specifying the name of the file.

Why File Handling is Required?

File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.

In simple words, file handling means reading and writing data to a file.

```

// Importing File Class
import java.io.File;
class GFG
{
    public static void main(String[] args)
    {

        // File name specified
        File obj = new File("myfile.txt");
    }
}

```

```
        System.out.println("File Created!");  
    }  
}
```

Output:

File Created!

File Operations in Java

In Java, a File is an abstract data type. A named location used to store related information is known as a File. There are several File Operations like creating a new File, getting information about File, writing into a File, reading from a File and deleting a File. Before understanding the File operations, it is required that we should have knowledge of Stream and File methods. If you have knowledge about both of them, you can skip it.

Stream

A series of data is referred to as a stream. In Java, Stream is classified into two types, i.e., Byte Stream and Character Stream.

- **Byte Stream:** Byte Stream is mainly involved with byte data. A file handling process with a byte stream is a process in which an input is provided and executed with the byte data.
- **Character Stream:** Character Stream is mainly involved with character data. A file handling process with a character stream is a process in which an input is provided and executed with the character data.

We can perform the following operation on a file:

- Create a File
- Get File Information
- Write to a File
- Read from a File
- Delete a File

Create a File

Create a File operation is performed to create a new file. We use the `createNewFile()` method of `File`. The `createNewFile()` method returns `true` when it successfully creates a new file and returns `false` when the file already exists. Let's take an example of creating a file to understand how we can use the `createNewFile()` method to perform this operation.

Example:

CreateFile.java

// Importing File class

import java.io.File;

// Importing the IOException class for handling errors

import java.io.IOException;

class CreateFile

{

 public static void main(String args[])

 {

 try

 {

 // Creating an object of a file

 File f0 = new File("D:FileOperationExample.txt");

 if (f0.createNewFile())

 {

 System.out.println("File " + f0.getName() + " is created successfully.");

 }

 else

 {

 System.out.println("File is already exist in the directory.");

 }

 }

 } catch (IOException exception)

 {

 System.out.println("An unexpected error is occurred.");

```

        exception.printStackTrace();
    }
}

```

Output:

File Operations in Java

File Operations in Java

Explanation: In the above code, we import the File and IOException class for performing file operation and handling errors, respectively. We create the f0 object of the File class and specify the location of the directory where we want to create a file. In the try block, we call the createNewFile() method through the f0 object to create a new file in the specified location. If the method returns false, it will jump to the else section. If there is any error, it gets handled in the catch block.

Write to a File

The next operation which we can perform on a file is "writing into a file". In order to write data into a file, we will use the FileWriter class and its write() method together. We need to close the stream using the close() method to retrieve the allocated resources. Let's take an example to understand how we can write data into a file.

Example:

WriteToFile.java

```

// Importing the FileWriter class
import java.io.FileWriter;
// Importing the IOException class for handling errors
import java.io.IOException;
class WriteToFile
{

    public static void main(String[] args)
    {

```

```

        try
        {
            FileWriter fwrite = new FileWriter("D:FileOperationExample.txt");
            // writing the content into the FileOperationExample.txt file
            fwrite.write("A named location used to store related information is
referred          to as a File.");
            // Closing the stream
            fwrite.close();
            System.out.println("Content is successfully wrote to the file.");
        }
        catch (IOException e)
        {
            System.out.println("Unexpected error occurred");
            e.printStackTrace();
        }
    }
}

```

Output:

File Operations in Java

File Operations in Java

Explanation: In the above code, we import the `java.io.FileWriter` and `java.io.IOException` classes. We create a class `WriteToFile`, and in its main method, we use the try-catch block. In the try section, we create an instance of the `FileWriter` class, i.e., `fwrite`. We call the `write` method of the `FileWriter` class and pass the content to that function which we want to write. After that, we call the `close()` method of the `FileWriter` class to close the file stream. After writing the content and closing the stream, we print a custom message. If we get any error in the try section, it jumps to the catch block. In the catch block, we handle the `IOException` and print a custom message.

Read from a File

The next operation which we can perform on a file is "read from a file". In order to write data

into a file, we will use the Scanner class. Here, we need to close the stream using the close() method. We will create an instance of the Scanner class and use the hasNextLine() method nextLine() method to get data from the file. Let's take an example to understand how we can read data from a file.

Example:

ReadFromFile.java

```
// Importing the File class
import java.io.File;
// Importing FileNotFoundException class for handling errors
import java.io.FileNotFoundException;
// Importing the Scanner class for reading text files
import java.util.Scanner;
class ReadFromFile
{
    public static void main(String[] args)
    {
        try
        {
            // Create f1 object of the file to read data
            File f1 = new File("D:FileOperationExample.txt");
            Scanner dataReader = new Scanner(f1);
            while (dataReader.hasNextLine()) {
                String fileData = dataReader.nextLine();
                System.out.println(fileData);
            }
            dataReader.close();
        }
        catch (FileNotFoundException exception)
        {
            System.out.println("Unexpected error occurred!");
            exception.printStackTrace();
        }
    }
}
```

```
}
```

Output:

File Operations in Java

Explanation: In the above code, we import the "java.util.Scannner", "java.io.File" and "java.io.IOException" classes. We create a class ReadFromFile, and in its main method, we use the try-catch block. In the try section, we create an instance of both the Scanner and the File classes. We pass the File class object to the Scanner class object and then iterate the scanner class object using the "While" loop and print each line of the file. We also need to close the scanner class object, so we use the close() function. If we get any error in the try section, it jumps to the catch block. In the catch block, we handle the IOException and print a custom message.

Delete a File:

The next operation which we can perform on a file is "deleting a file". In order to delete a file, we will use the delete() method of the file. We don't need to close the stream using the close() method because for deleting a file, we neither use the FileWriter class nor the Scanner class. Let's take an example to understand how we can write data into a file.

Example:

DeleteFile.java

```
// Importing the File class
```

```
import java.io.File;
```

```
class DeleteFile
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        File f0 = new File("D:FileOperationExample.txt");
```

```
        if (f0.delete())
```

```
        {
```

```
            System.out.println(f0.getName()+ " file is deleted successfully.");
```

```
        }
```

```
    else
```

```
    {
```



```
        System.out.println("Unexpected error found in deletion of the file.");
    }
}
}
```

Output:

File Operations in Java

Explanation: In the above code, we import the File class and create a class DeleteFile. In the main() method of the class, we create f0 object of the file which we want to delete. In the if statement, we call the delete() method of the file using the f0 object. If the delete() method returns true, we print the success custom message. Otherwise, it jumps to the else section where we print the unsuccessful custom message.

File Streams in Java

In Java, a sequence of data is known as a stream. This concept is used to perform I/O operations on a file. You can create a file stream from the filename, a File object, or a FileDescriptor. object. Use file streams to read data from or write data to files on the file system.

There are two types of streams in java:

1. File Input Stream
2. File Output Stream

1. File Input Stream:

The Java InputStream class is the superclass of all input streams. The input stream is used to read data from numerous input devices like the keyboard, network, etc. InputStream is an abstract class, and because of this, it is not useful by itself. However, its subclasses are used to read data. There are several subclasses of the InputStream class, which are as follows:

- AudioInputStream
- ByteArrayInputStream
- FileInputStream
- FilterInputStream
- StringBufferInputStream
- ObjectInputStream

Creating an Input Stream:

Syntax:

```
// Creating an InputStream
```

```
InputStream obj = new FileInputStream();
```

Here, an input stream is created using FileInputStream.

Note: We can create an input stream from other subclasses as well as InputStream.

Methods of Input Stream:

1	<code>read()</code>	Reads one byte of data from the input stream.
2	<code>read(byte[] array)()</code>	Reads byte from the stream and stores that byte in the specified array.
3	<code>mark()</code>	It marks the position in the input stream until the data has been read.
4	<code>available()</code>	Returns the number of bytes available in the input stream.
5	<code>markSupported()</code>	It checks if the <code>mark()</code> method and the <code>reset()</code> method is supported in the stream.
6	<code>reset()</code>	Returns the control to the point where the mark was set inside the stream.
7	<code>skip()</code>	Skips and removes a particular number of bytes from the input stream.
8	<code>close()</code>	Closes the input stream.

2. Output Stream:

The output stream is used to write data to numerous output devices like the monitor, file, etc. OutputStream is an abstract superclass that represents an output stream. OutputStream is an abstract class and because of this, it is not useful by itself. However, its subclasses are used to write data. There are several subclasses of the OutputStream class which are as follows:

- ByteArrayOutputStream
- FileOutputStream
- StringBufferOutputStream
- ObjectOutputStream
- DataOutputStream
- PrintStream

Creating an Output Stream

Syntax:

```
// Creating an OutputStream
```

```
OutputStream obj = new FileOutputStream();
```

Here, an output stream is created using FileOutputStream.

Note: We can create an output stream from other subclasses as well as OutputStream.

Methods of Output Stream:

1. write() Writes the specified byte to the output stream.
2. write(byte[] array) Writes the bytes which are inside a specific array to the output stream.
3. close() Closes the output stream.
4. flush() Forces to write all the data present in an output stream to the destination.



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

The Motto of the University

(SEWA)

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



Bachelor of Computer Applications (BCA)

Course : Computer Graphics

Course Code: BCA-4-04T

ADDRESS: C/28, THE LOWER MALL, PATIALA-147001

WEBSITE: www.psou.ac.in

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU, ALL COPYRIGHTS WITH JGND PSOU, PATIALA



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA

(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME COORDINATOR :

Dr. Monika Pathak

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

PROGRAMME CO-COORDINATOR & COURSE COORDINATOR :

Dr. Gaurav Dhiman

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

Bachelor of Computer Applications (BCA)
BCA-4-04T: Computer Graphics

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 40%

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Course: Computer Graphics	
Course Code: BCA-4-04T	
Course Outcomes (COs) After the completion of this course, the students will be able to:	
CO1	Understand the basics of computer graphics, different graphics systems and applications of computer graphics.
CO2	Demonstrate proficiency in 2D graphics programming, including concepts like 2D transformations.
CO3	Analyze and implement key computer graphics algorithms, such as line drawing algorithms, polygon filling algorithms, and clipping algorithms.
CO4	Extract scene with different clipping methods and its transformation to graphics

	display device.
CO5	Explore projections and visible surface detection techniques for display of 3D scene on 2D screen.

Detailed Contents:

Module	Module Name	Module Contents
Module 1	Introduction to Computer Graphics	Applications areas, Components of Interactive Computer Graphics System. Video Display Devices: Refresh cathode ray tube systems – raster scan CRT displays, random scan CRT displays, colour CRT-monitors, direct view storage tube. Flat panel displays – emissive vs non emissive displays, LCD displays, plasma panel displays, 3-D viewing devices, virtual reality.
Module II	Scan conversion and 2D Graphics	Scan converting a Point, Line (Direct, DDA and Bresenham line algorithms), Circle (Direct, Polar, Bresenham and Mid-point circle algorithms), Ellipse (Direct, Polar and Midpoint ellipse algorithms), Area filling techniques (Boundary fill, Flood fill, scan line area fill algorithm), Limitations of scan conversion. 2D Cartesian and Homogeneous co-ordinate system, Geometric transformations (Translation, Scaling, Rotation, Reflection and Shearing), Composite transformations, 2D dimensional viewing transformation and clipping (Cohen – Sutherland, Liang-Barsky, Sutherland-Hodgeman algorithms).
Module III	3D Graphics	3D Cartesian and Homogeneous co-ordinate system, Geometric transformations (Translation, Scaling, Rotation, Reflection), Composite transformations. Mathematics of Projections: Perspective Projections - Mathematical Description and Anomalies of perspective projections. Parallel Projections – Taxonomy of Parallel Projections and their Mathematical Description. Introduction to 3D viewing pipeline and 3D clipping.
Module IV	Hidden surface elimination algorithms	Z-buffer, scan-line, sub-division, Painter's algorithm. Illumination Models: Diffuse reflection, Specular reflection, refracted light, texture surface patterns, Halftoning, Dithering. Surface Rendering Methods: Constant Intensity method, Gouraud Shading, Phong Shading.

Books

1. R.A. Plastock and G. Kalley, “Computer Graphics”, McGraw Hill.
2. Donald Hearn and M. Pauline Baker, “Computer Graphics”, Pearson Education.
3. J.D. Foley, A.V. Dam, S.K. Feiner, J.F. Hughes,. R.L Phillips, “Introduction to Computer Graphics”, Addison Wesley Publishing.

Bachelor of Computer Applications (BCA)

Computer Graphics

Module I: Introduction to Computer Graphics
1.1 Applications areas.
1.2 Components of Interactive Computer Graphics System.
1.3 Video Display Devices: Refresh cathode ray tube systems – raster scan CRT displays, random scan CRT displays, colour CRT-monitors, direct view storage tube.
1.4 Flat panel displays – emissive vs non emissive displays, LCD displays, plasma panel displays, 3-D viewing devices, virtual reality.

OVERVIEW OF COMPUTER GRAPHICS

- Computer graphics deals with representation of information in pictorial form.
- It is a way of generating of images or any kind of object with the aid of computers.
- It plays an important role in many fields like education, photography, filming and many more.

Definitions:

- **Computer Graphics:** It is a field of computer science that deals with creating, manipulating, storing and displaying pictures or objects on a display device.
- **Pixel:** It is a smallest element on the display screen.
- **Resolution:** The number of pixels on a display monitor expressed in terms of number of pixels on horizontal axis on vertical axis is called as resolution.
- **Rasterization:** The process of determining appropriate pixels for representing picture or object on screen is called as rasterization.
- **Scan conversion:** The process of representing continuous picture or object as a collection of discrete pixel is called as scan conversion.

Advantages of Computer Graphics:

- Produces high quality graphics.
- Provides use of tools to generate and manipulate objects or pictures.
- Used to work with 2D and 3D objects.
- Used in animations.

COMPUTER GRAPHICS APPLICATIONS

- Computer graphics can be used in various fields. Some of the common applications are Computer Aided Design, Computer Art, Entertainment and Education and Training

Computer Aided Design:

- Design processes of architecture and engineering systems.
- Design of automobiles, textiles, spacecraft, aircrafts, buildings, etc.

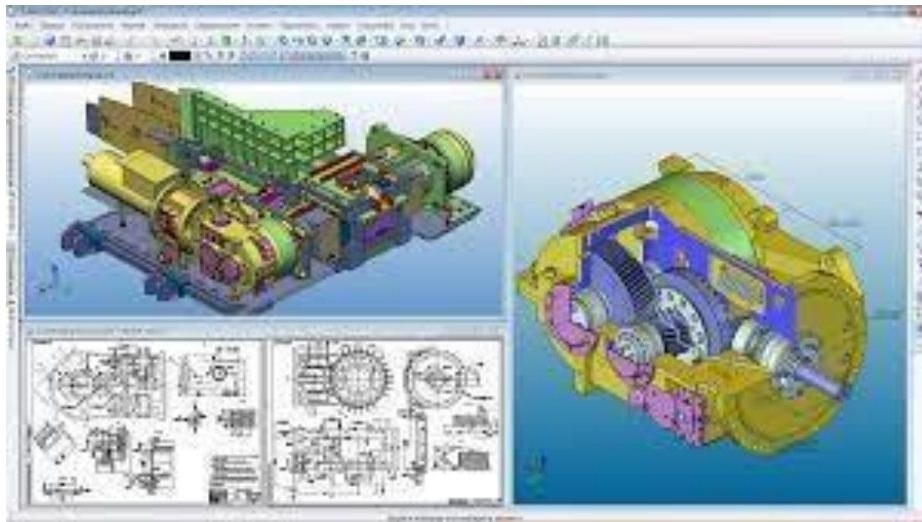


Fig. 1 Use of CAD in computer graphics

Computer Art:

- It is majorly used in the field of fine arts and commercial applications.
- Use of applications like mathematics package, paint package, animation program and desktop publishing software helps in computer art. E.g., cartoon drawing, painting.



Fig. 2 Use of Computer Art in computer graphics

Entertainment:

- It can be used to create any motion pictures and videos.
- Animated movies or video clips can be created.
- Sound and music can be used to generate any movie clips, animated shows or TV shows.
- Games can also be made with interactivity features.



Fig. 3 Use of Entertainment in computer graphics

Education and Training :

- It is used in classroom to enhance educational experience by simplifying teacher's work.
- Models can be generated which can be useful for teaching and making understanding more easier.
- In distance learning, where students are not at the same place, use of computer graphics plays an important role, by means of conducting tests. E.g., kahoot.it, quizzes.com.
- For training, simulators can be made as hands on experience for candidates for better understanding.

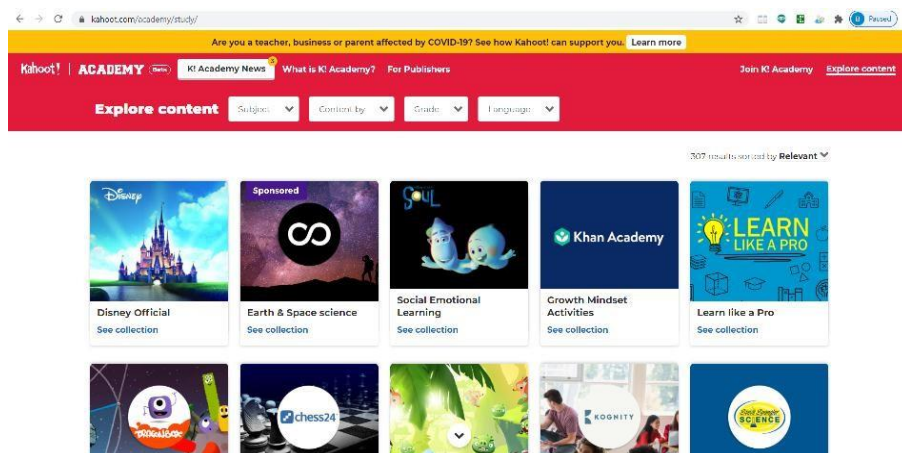


Fig. 4 Use of Kahoot in Education

Presentation Graphics:

- Reports can be generated on slides.
- It is used for summarizing mathematical, financial, statistical and scientific data.
- Graphs and charts can be generated to make any kind of presentation. E.g., pie charts, bar charts.



Fig. 5 Use of Presentation graphics in computer graphics

Visualization:

- To check insights of data, to study behavior of processes, visualization is required with proper use of computer graphics.



Fig. 6 Use of Visualization in computer graphics

Image Processing:

- Used in applications where images or photographs requires editing.
- Processing on existing images done for more enhancements.



Fig. 7 Use of Image Processing in computer graphics

Graphical User Interfaces :

- A window manager is an important component in GUI that allows a user to display multimedia window areas.
- For fast selection processing, use of menus, icons and buttons are made in an interface.



Fig. 8 Use of GUI in computer graphics

COMPUTER GRAPHICS SOFTWARE

- Graphics software can be classified into two categories:
 - a. General Programming Packages
 - b. Special Purpose Applications Packages

a. General Programming Packages

- They are like programming languages; they contain a rich set of graphics functions.
- Basic functions include:
 - Generate picture components like circles, polygons, straight lines, etc.

- Filling colors and setting intensity values
- Selecting views
- Applying transformations
- E.g., graphics library system, Open GL, Open CV

b. Special Purpose Application Packages

- These set of graphics packages are designed for non-programmers.
- Users can create their own graphics without worrying about the underlying concepts.
- The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms.
- E.g., Paint, CAD, etc.

Coordinate Representation:

- General graphics packages are generally designed to be used with Cartesian coordinate specifications.
- The coordinate values for an object should be converted to Cartesian coordinates if they were specified in some other reference frame, before they are passed as input to a graphics package.
- Special purpose packages allow the use of other coordinate systems based upon the applications used.
- Various Cartesian reference frames can be used to construct and display a scene.
- Individual shape of an object can be constructed in a scene within separate coordinate reference frames called modelling coordinates or local coordinates or master coordinates.
- Objects can be placed into appropriate positions within a scene using a reference frame called as world coordinates, once the individual object shapes have been specified.
- For display, the world coordinate description of the scene is transferred to one or more output device reference frames, called as device coordinates or screen coordinates.
- Before final conversion to specific device coordinates, a graphics system first converts world coordinates positions to normalized device coordinates in the range from 0 to 1.
- This is done to ensure that the system is independent of any device used as a workstation.

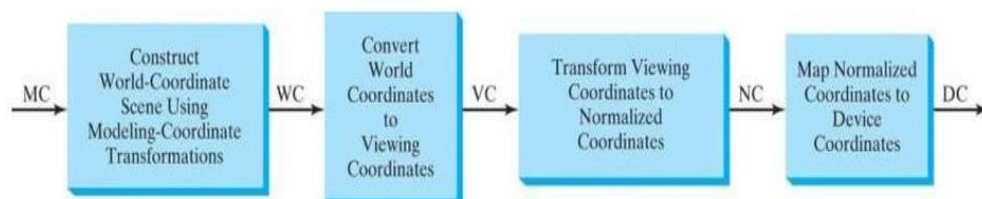


Fig. 9 Viewing Pipeline

Graphics Functions:

- Users are provided with various functions for creating and manipulating pictures by general-purpose graphics package.
- **Output primitives:** These are the basic building blocks for pictures. It includes string and entities like points, straight lines, curved lines, circles and polygons.
- **Attributes:** The properties of output primitives are referred to as attributes that describes how a primitive is to be displayed. It includes line styles, color specifications, intensity of colors and text styles.
- **Geometric transformations:** It is used to change the size, orientation or position of an object within a scene.
- **Viewing transformations:** It is used to specify view that is to be presented and portion of the output display area that is to be used.
- **Input functions:** It is used to control and process the data flow from interactive devices.
- **Control operations:** It is used in clearing a display screen or initializing parameters.

Software Standards:

- The primary intention of standardization of graphics software is portability.
- The packages are designed with standard graphics functions so they can be used with different hardware systems and used in different implantation and applications.
- The **Graphical Kernel System (GKS)** was developed to adopt the first graphics software standard by the International Standard Organization (ISO).
- Another graphics standard developed and approved was **Programmers Hierarchical Interactive Graphics System (PHIGS)**, which is an extension of GKS.
- It provides improved capabilities for object modelling, color specifications, surface rendering and picture manipulation.
- An extension of PHIGS, called as PHIGS+, was developed to provide 3D surface shading capabilities.
- PHIGS does not provide a standard methodology for graphics interface to output devices and does not specify methods for storing and transmitting pictures.
- **Computer Graphics Interface (CGI):** system provides

standards for device interface methods and the **Computer Graphics Metafile (CGM)** system specifies standards for archiving and transporting pictures.

INPUT DEVICES

Keyboards:

- To enter any kind of string or number, an alphanumeric keyboard is used as an input in a graphics system.
- A general keyboard can consist of 84 keys or 101 keys.
- A keyboard consists of numeric keys (0 to 9), alphabetic keys (A to Z), control keys (Ctrl, Shift, Alt), special symbol keys (!, @, #, ", ?), cursor control keys (arrow keys), function keys (F1 to F9).
- Keyboards are not only used to enter any numbers, or symbols or alphabets, it is also be used for menu selection.
- A numeric keypad is on the right of keyboard for fast entry of numeric data.



Fig. 10 Keyboard

Mouse:

- To position the cursor on the screen, mouse is used.
- The roller present at the bottom of the mouse records the direction of the movement.
- The motion of the mouse can also be detected by optical sensor in which the mouse is moved over a special mouse pad that has a grid of horizontal and vertical lines.
- The optical sensor detects movement across the lines in the grid.
- The mouse consists of two or three buttons on the top to execute any kind of operation.
- There is a type of mouse called as Z mouse which can be used to pick an object, rotate that object, move the object in any direction and navigate our view through a three dimensional scene.



Fig. 11 Mouse

Trackball and Spaceball:

- A trackball is a ball that is used to rotate with palm of the hand or with the fingers, to produce screen-cursor movement.
- A resistor with sliding or rotating contact (called potentiometers) attached to the ball, is used to measure the amount and direction of rotation.
- Trackballs are generally mounted on keyboards and is a two-dimensional positioning device.
- A spaceball does not actually move.
- A sensor whose resistance varies when force applied (called strain gauges) measures the amount of pressure applied to the spaceball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions.
- Spaceballs are used for three-dimensional positioning and selection operations in CAD, animation, virtual-reality systems, modeling and other applications.



Fig. 12 Trackball and Spaceball

Joysticks:

- A joystick consists of a small, vertical lever (called the stick) mounted on a base that is used to move the screen cursor in all directions.
- Position on the screen is selected with actual stick movement or by applying pressure on stick.
- Joysticks can be standalone device or can be mounted on a keyboard.
- The direction of the screen-cursor movement depends upon the direction in which the stick is moved.
- Potentiometers mounted at the base of the joystick measure the amount of movement, and springs return the stick to the center position when it is released.



Fig. 13 Joystick

Data Glove:

- Data glove is used to grasp a virtual object.
- The glove is made up of many sensors that helps to detect motions of fingers and hand.
- The information about the position and orientation of hand is provided by the electromagnetic coupling between transmitting and receiving antennas.
- Input from the glove can be used to position or manipulate objects in a virtual scene.
- A two-dimensional projection of the scene can be viewed on a video monitor.



Fig. 14 Data Glove

Image Scanner :

- Drawings, graphs, color and black-and-white photos, or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.
- Once we have the internal representation of a picture, we can apply transformations to rotate, scale, or crop the picture to a particular screenarea.
- For scanned text input, various editing operations can be performed onthe stored documents.
- Some scanners are able to scan either graphical representations or text,and they come in a variety of sizes and capabilities.



Fig. 15 Image Scanner

Touch Panels :

- A type of display screen that has a touch-sensitive transparent panelcovering the screen are called touch panels.
- The input is recorded through finger when in contact with the screen.
- It allows a user to interact directly what is being displayed instead ofusing mouse and keyboards.
- Generally used on smart phones and laptops.



Fig. 16 Touch Panel

Light Pens:

- It is a pencil-shaped device used to select screen positions by detecting the light coming from points on the CRT screen.
- Also used to select menu driven applications or draw pictures on monitor screen.
- It consists of a photocell and optical system placed in a small tube.
- When its tip is moved over the monitor screen, the photocell-sensing element detects screen location and sends corresponding signals to CPU.



Fig. 17 Light Pens

Voice Systems:

- It is used in some graphics workstations as input devices to accept voice commands.
- These systems operate by matching an input against a predefined dictionary of words and phrases.
- It is easy to use and most efficient device. E.g., a microphone.



Fig. 18 Voice Systems

DISPLAY DEVICES

- There are number of display devices used in graphics.
- Output devices is another name of display devices.
- The display devices generally used are CRT and LCD.

Storage Tubes Graphics Displays:

- This is also known as Direct View Storage Tubes (DVST).
- The raster scan display refreshes the screen to maintain a screen image whereas DVST gives alternate method of maintaining the screen image without refreshing.

Components of DVST:

- i) Two Electron Guns
- ii) Phosphor coated screen,
- iii) Storage grid mesh
- iv) Collector

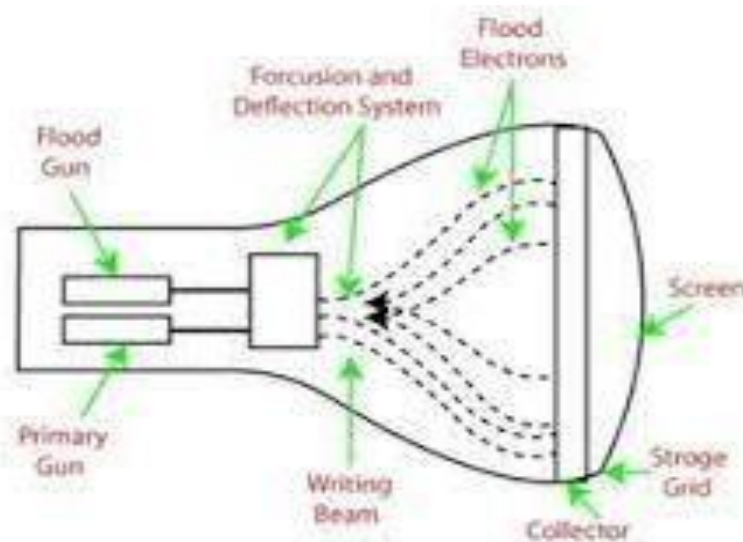


Fig 19. Working of Direct View Storage

Tube Working Principle:

1. There are two electron guns, namely primary gun and flood gun.
2. The primary gun stores the picture pattern and the flood gun maintains the picture display.
3. A primary gun produces high-speed electrons, which strike on the storage grid to draw the picture pattern.
4. Continuous low speed electrons from flood gun pass through the control grid and got attracted to the positive charged areas of the

storage grid.

5. The low speed electron then penetrates the storage grid and strike the phosphor coating.
6. During this process, the collector behind the storage grid smooths out the flows of food electrons.

Merits:

- It has flat screen.
- Refreshing of CRT is not required.
- Complex picture can be displayed on high resolution.

Demerits:

- The performance of DVST is poor compared to CRT.
- Erasing requires removal of charge on the storage grid.
- Selective or part erasing of screen is not possible.
- Erasing of screen produces flash over the entire screen.

Calligraphic Refresh Graphics Displays:

Calligraphic refresh graphic displays are also referred as random scan or stroke-writing or vector displays.

Components:

A CRT display unit with electron beam,, Refresh display files,Graphics controller

Working Principle:

- A CRT, as a random scan display unit, has an electron beam directedonly to the parts of the screen where a picture is to be drawn.
- Random scan monitors draw a picture one line at a time.
- The component of a picture can be drawn and refreshed by a random scan system in a specified order.
- A picture definition is stored as a set of line-drawing commands in an area of memory called a refresh display file or refresh buffer.
- To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line one by one.
- After all line drawing commands have been processed, the system cycles back to the first line command in the list and repeat the procedure of scan, display, and retrace.

Example:

- A pen plotter

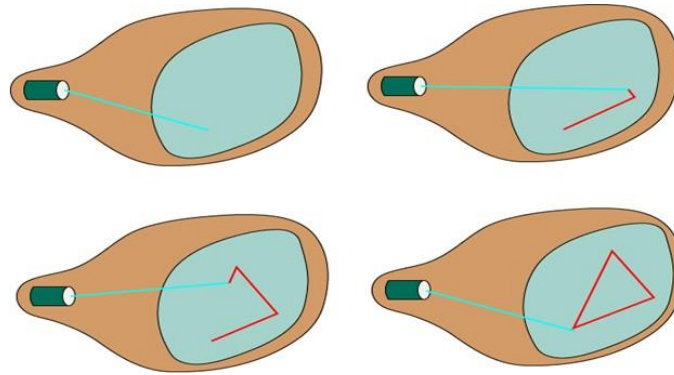


Fig 20. Working of Random Scan Systems

- Random scan displays draw all the component lines of a picture with 30 to 60 times for each second.
- High quality vector systems are capable of handling approximately 100,000 short lines at this refresh rate.
- The faster refreshing of the set of lines could burn out the phosphor. Therefore, when a small set of lines are to be displayed, each refresh cycle is delayed to avoid greater refresh rates, typically 60 frames per second.
- Random scan systems are designed for line drawing applications; hence it cannot display realistic shaded scenes.
- Vector displays have higher resolution than raster systems, as picture definition is stored as a set of line drawing instructions instead of a set of intensity values for all screen points.
- These vector displays produce smooth line drawings, because the CRT beam directly follows the line path.

Raster Refresh Graphics Displays:

A raster scan display is the most common method of drawing images on a CRT screen.

Working Principle:

- In this method, horizontal and vertical deflection signals are generated to move a beam all over the screen in a pattern for displaying any image.
- The electron beam is swept across the screen one row at a time from top to bottom.
- The electron beam sweeps back and forth from left to right across the screen.

Horizontal Retrace

While electron beam moves from left to right the beam is on and when it moves back from right to left the beam is off. This phenomenon is known as horizontal retrace.

Vertical Retrace

As soon as the beam reaches the bottom of the screen, it is turned off and is rapidly retraced back to the top to start again. This is known as vertical retrace.

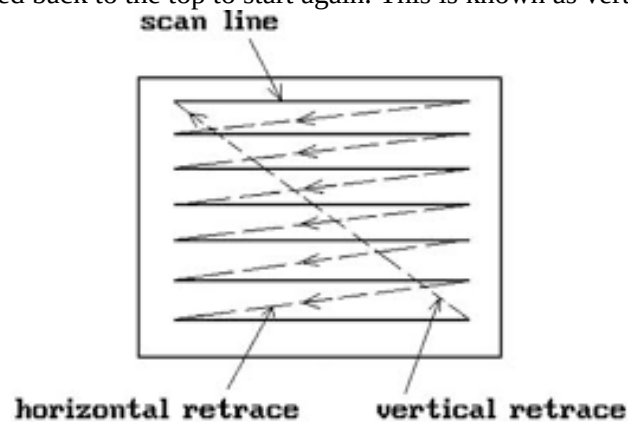


Fig 21. Horizontal and vertical retrace

- Raster scan displays maintain the steady image on the screen by repeating scanning of the scan image. This process is known as refreshing the screen.

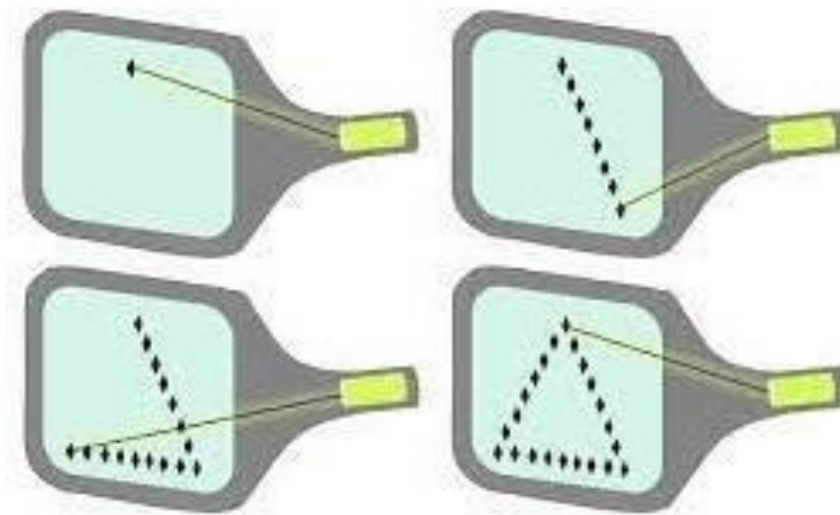


Fig 22. Working of Raster Scan Systems

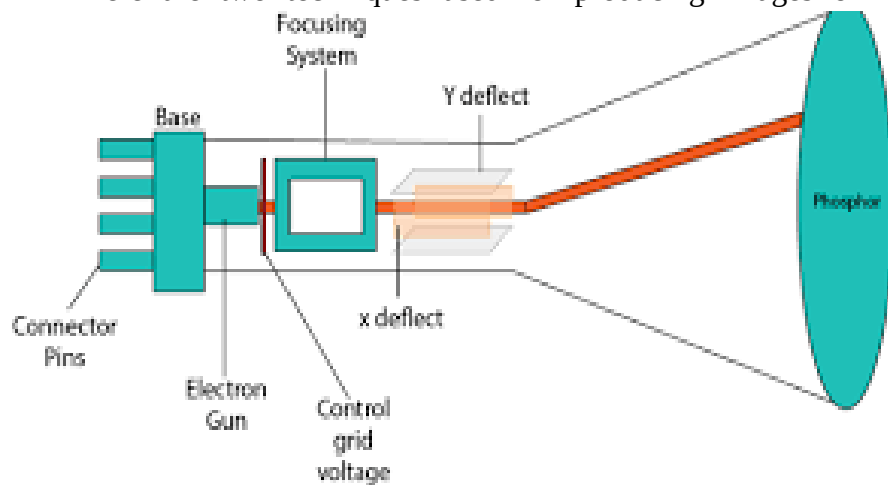
Random Scan Display V/S Raster Scan Display:

Random Scan Display	Raster Scan Display
It has high resolution.	It has less resolution.
The electron beam is directed on portion of screen where picture is to be displayed.	The electron beam is directed to the whole screen i.e. scanning one row at a time.
It is expensive.	It is less expensive.

It stores picture definition as a set of line commands called refresh display file.	It stores picture definition in refresh buffer or frame buffer.
Smooth line is produced.	Zigzag line is produced.
Hard to display realistic image.	Realistic image is displayed.
It uses line drawing applications.	It uses pixels along scan line to draw an image.
Refresh rate is 30 to 60 frames/second.	Refresh rate is 60 to 80 frames/second.
Memory required is less.	Memory required is more.

Cathode Ray Tube (CRT) :

- A CRT is an evacuated glass tube.
- An electronic gun at the rear of the tube produces a beam of electron, which is directed towards the front of the screen by a high voltage.
- The inner side of the screen is coated with phosphor substance, which gives off light when it is stroked by electron.
- The control grid voltage determines the velocity achieved by the electrons before they hit the phosphor.
- The control grid voltage determines how many electrons are actually in the electron beam.
- Thus, control grid controls the intensity of the spot where the electron beam strikes the screen.
- The focusing system concentrates the electron beam so that the beam converges to a small point when it hits the phosphor.
- The deflection system of CRT consists of two pairs of parallel plates, namely horizontal and vertical deflection plates.
- The voltage applied to vertical deflection of the electron beam and the voltage applied to the horizontal deflection plates controls the horizontal deflection of the electron beam.
- There are two techniques used for producing images on CRT



Cathode Ray Tube

screen, namely Random Scan and Raster Scan.

Fig 23. Working of CRT Monitors

Color CRT:

- A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light.
- It generates a range of colors by combining the emitted light from different phosphors.
- There are two basic techniques used for producing color displays:
 - Beam-penetration technique
 - Shadow mask technique

Beam-Penetration Techniques:

- This technique is used with random-scan monitors.
- In this technique, the inside of CRT screen is coated with two layers of phosphor, usually red and green.
- The displayed color depends on how far the electron beam penetrates into the phosphor layers.
- The outer layer is of red phosphor and the inner layer is of green phosphor.
- A beam of slow electrons excites only the outer red color.
- A beam of very fast electrons penetrates through the red layer and excites the inner green layer.
- At intermediate beam speeds, combination of red and green light are emitted and two additional colors, orange and yellow are displayed.
- The beam acceleration voltage controls the speed of electrons and hence the screen color at any point on the screen.

Merits:

- It is inexpensive technique to produce color in random scan monitors.

Demerits:

- It can display only four colors.
- The quality of picture produced by this technique is not good as compared to other techniques.

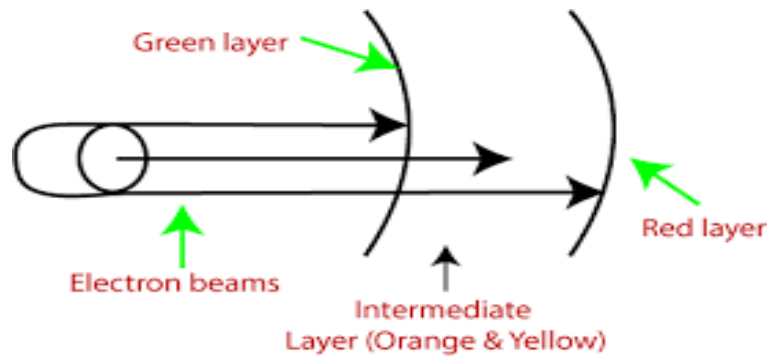


Fig 24. Beam Penetration Technique

Shadow Mask Technique:

- The shadow mask technique produces a much wider range of colors than the beam penetration technique.
- Hence, this technique is commonly used in raster-scan displays including color TV.
- In a shadow mask technique, CRT has three phosphor color dots at each pixel position.
- One phosphor dot emits a red light, second emits green light and third emits blue light.
- It has three electron guns, one for each color dot and a shadow mask grid just behind the phosphor-coated screen.
- The shadow mask grid consists of series of holes aligned with the phosphor dot pattern.

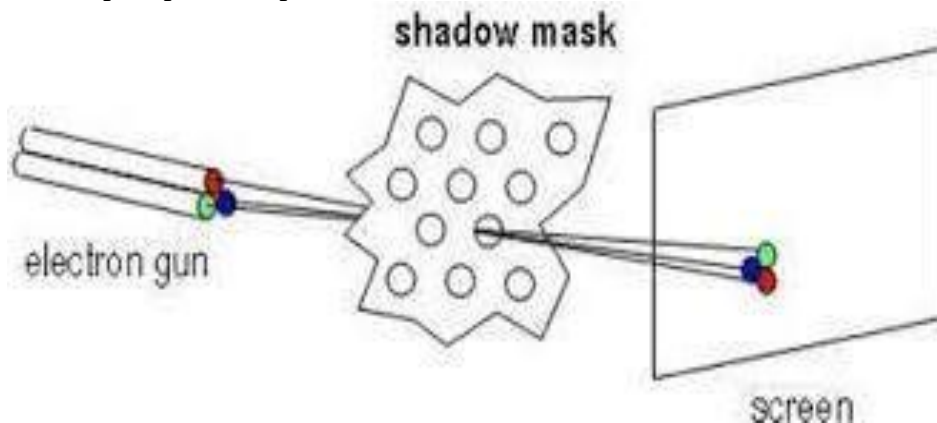


Fig 25. Shadow Mask Technique

Liquid Crystal Displays (LCD):

- The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid.
- Two glass plates, each containing a light polarizer at the right angles to the other plate, squeeze the liquid-crystal material.
- Rows of horizontal transparent conductors are built into one glass plate and columns of vertical conductors are put into the other.

plate.

- The intersection of two conductors defines a pixel position.
- In the ON state, polarized light passing through material is twisted so that it will pass through the opposite polarizer.
- It is then reflected back to the viewer.
- To turn OFF the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that light is not twisted.

Merits:

- Less bulky.
- Suitable component for portable devices like laptops, mobile phones, etc.
- Power consumption is much less so ideal to be operated by batteries.

Demerits:

- More expensive than the CRT and more difficult to manufacture.

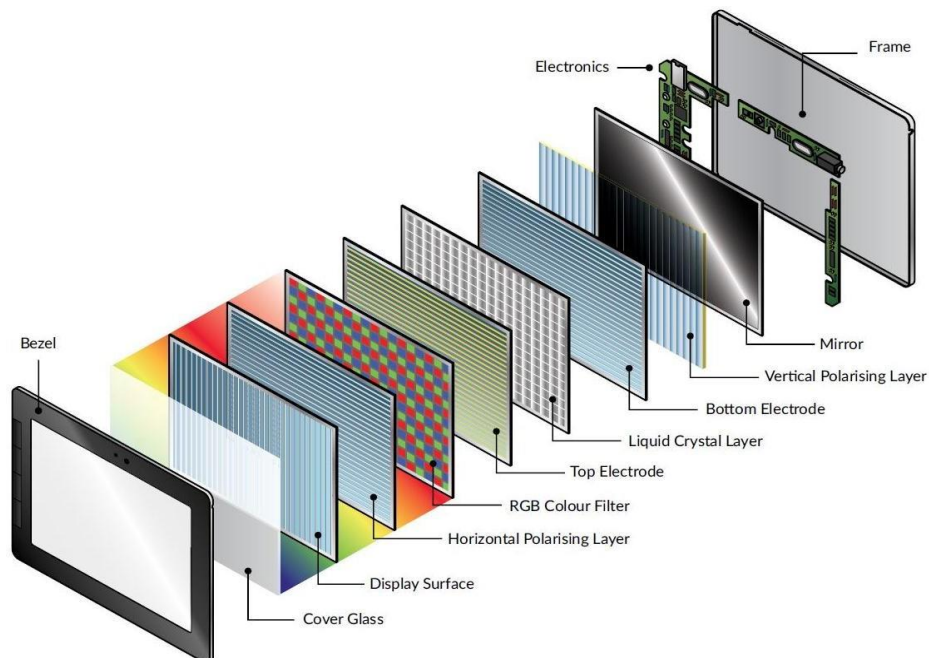


Fig 26. Working of LCD

LCD Displays V/S CRT Monitors:

LCD Displays	CRT Monitors
Thin, compact, lightweight. Takes up less space.	Bulky, heavier and takes up more space.
Lack in brightness than CRT.	Appreciable amount of brightness.

Consumes less electricity and produce less heat.	Consumes more power and produce more heat.
Much expensive.	Less expensive.
Portable.	Not portable.
Restricted viewing angles. It needs to be viewed from front to have better view.	Viewable from almost every angle.

RASTER SCAN SYSTEMS

- Hardware architecture of the raster scan system is shown in the Fig 9.
- In addition to CPU, a special-purpose processor, called video controller or display controller, is used to control the operation of the display device.
- In this architecture, frame buffer is allocated a dedicated area in system memory.
- Frame buffer can be anywhere in the system memory.

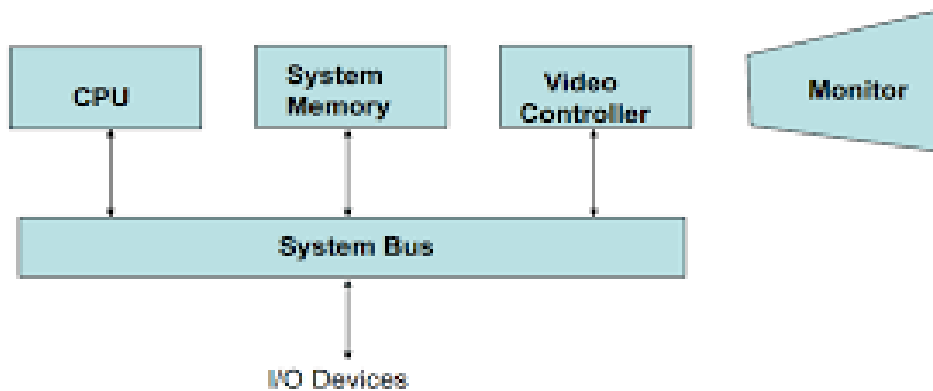


Fig 27. Architecture of a Raster Scan System

Video Controller:

- Another variant of a raster scan system is shown in Fig 10.
- A dedicated memory area is allocated to the frame buffer.
- Video controller is given direct access to frame buffer.
- The organization can render scene quickly because video controller do not have to put a request on system bus to read intensity value from frame buffer.
- In some system, origin is set at the bottom left location of the screen.

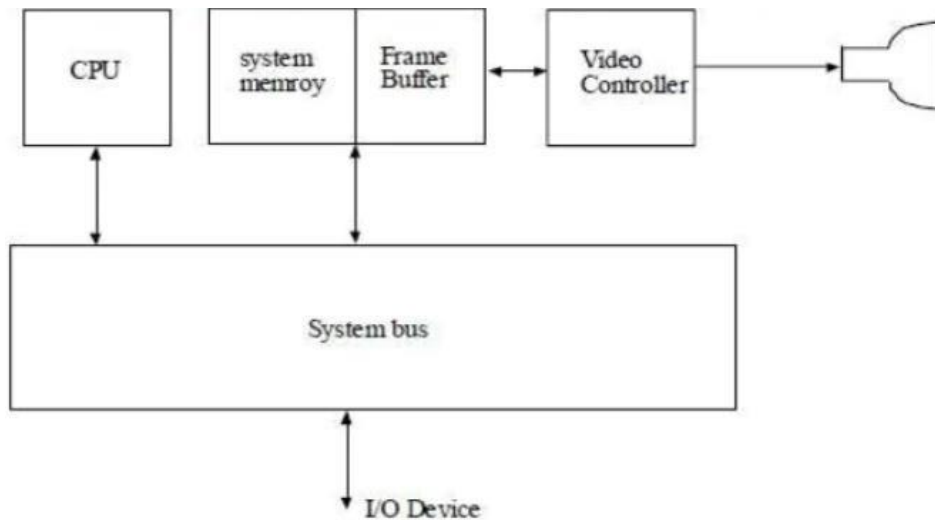


Fig 28. Architecture of a Raster Scan System

Display Processor:

- Display processor is also known as graphics controller or a displaycoprocessor.
- Its purpose is to free the CPU from graphics operations.
- The major task of a display processor is to digitize a picture definition given in an application program into a set of pixel intensity values for storage in frame buffer.
- This digitize process is called scan-conversion.
- Graphics commands specify straight lines and other geometric objectsare scan converted into a set of discrete intensity points.
- Scan converting a straight line means, we have to locate the pixel positions closest to the line path and store the intensity for each positionin the frame buffer.

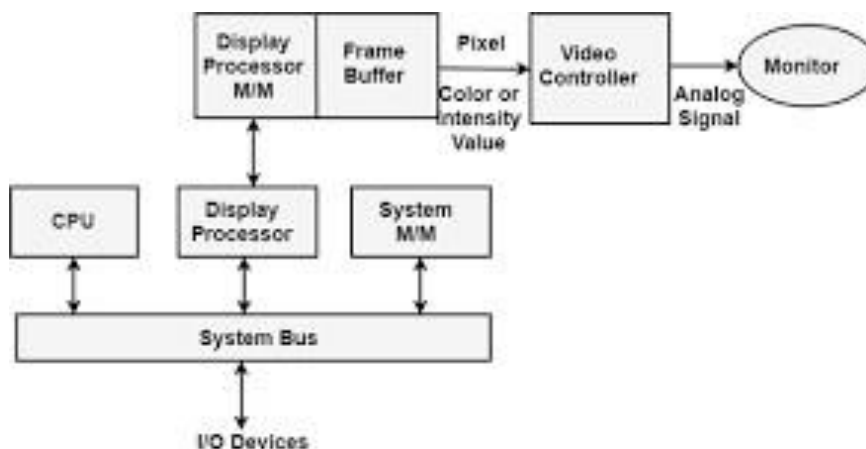


Fig 30. Raster Scan System with a display processor

RANDOM SCAN SYSTEMS

- Hardware architecture of random scan system is shown in Fig 13.

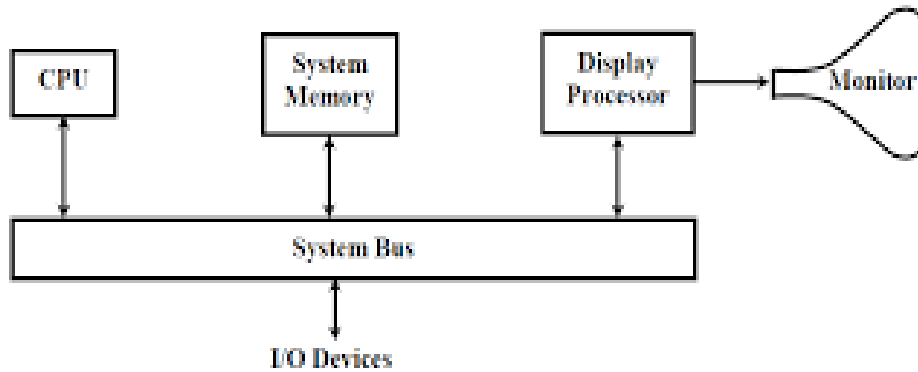


Fig 31. Random Scan System

- Application program resides in system memory.
- Graphics packages translate the graphics commands in the application program into a display file.
- Display file is also stored in system memory.
- Display processor accesses the content of display file and display on monitor screen.
- Display processor retrieves one by one command from display file and draw it on the screen.
- Display processor in random scan system is also called a graphics controller or display processing unit.
- Random scan devices render the scene using short line.
- Electron beam track the line position directly, rather than going through each and every pixel on screen.
- Entire scene is defined using segment of line in application program.

Module II: Scan conversion and 2D Graphics
2.1 Scan converting a Point.
2.2 Line (Direct, DDA and Bresenham line algorithms).
2.3 Circle (Direct, Polar, Bresenham and Mid-point circle algorithms).
2.4 Ellipse (Direct, Polar and Midpoint ellipse algorithms).
2.5 Area filling techniques (Boundary fill, Flood fill, scan line area fill algorithm).
2.6 Limitations of scan conversion.
2.7 2D Cartesian and Homogeneous co-ordinate system,
2.8 Geometric transformations (Translation, Scaling, Rotation, Reflection and Shearing).
2.9 Composite transformations.
2.10 2D dimensional viewing transformation and clipping (Cohen –Sutherland, Liang-Barsky, Sutherland-Hodge man algorithms).

INTRODUCTION OF SCAN CONVERSION

Scan conversion:

- Scan conversion is defined as a process of representing continuous graphic object as a collection of discrete pixels. It is also called as **Rasterization**.
- The graphics objects can be a point, line, circle, arc, rectangle, polygon and other shapes.
- Graphics objects are continuous.
- Pixels are always discrete. Each pixel has two states: on and off. 0 is addressed by pixel off and 1 is addressed by pixel on.
- The video display device of the computer has a circuit that converts the binary values of 0 and 1 into the information of pixel on and off.
- Image is represented as collection of pixels.

Pixel:

- Pixel is a smallest unit of a digital image or graphics object, which can be displayed and represented on a display device. A pixel is also called as a pel or picture element.
- A pixel is a basic logical unit in the graphics and the number of pixel are combined to form a complete image, video or text.
- A pixel is represented by a dot or a square on a display screen.
- Pixels are the basic building blocks of an image that are generated using geometric coordinates.

- Depending on the display monitor and graphics card, the size and color combination of pixel varies and is measured in terms of display resolution.

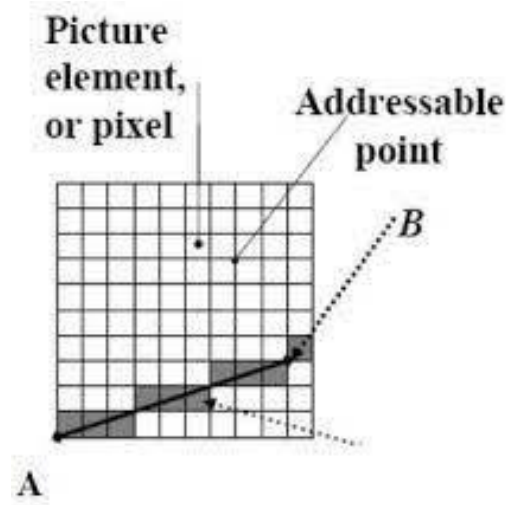


Fig 32. Pixel representation

SCAN CONVERSION OF POINTS

- Vectors are defined mathematically in an infinite, real number Cartesian Co-ordinate system.
- The physical coordinates of the pixel on the computer screen based upon the screen resolution is called as screen coordinates, device coordinates, or pixel coordinates.
- A display device deals with finite, discrete coordinates.
- Both x-coordinate and y-coordinate are positive values.
- On a display device, the origin (0, 0) is always on the top-left corner of the screen.

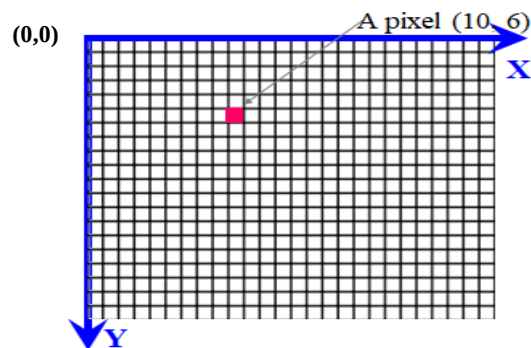


Fig 33. A pixel representation as screen coordinates on a display device

- The values in the x-coordinate increases from left to right and the values of the y-coordinate increases from top to bottom.

- A mathematical point (x, y) where x and y are real numbers within an image area, needs to be scan-converted to a pixel location at (x', y') .
- Each pixel on graphic display does not represent a mathematical pointlike $P(2.9, 4.22)$.
- This may be done by making x' and y' to be the integer part of x and y , by applying any mathematical function.

Example:

Floor is a function that returns largest integer that is less than or equal to the argument

$$x' = \text{Floor}(x)$$

$$y' = \text{Floor}(y)$$

for point $P(2.9, 4.22)$.

$$x' = \text{Floor}(2.9) \text{ and } y' = \text{Floor}(4.22),$$

$$(x', y') \text{ will be } (3, 4).$$

SCAN CONVERSION OF LINES

A line has two end points and is defined by an equation $y = mx + c$, where m is the slope of the line and c is the y intercept.

Consider the two end points as $P1(x1, y1)$ and $P2(x2, y2)$. The equation of the line is used to determine the x and y coordinates of all the points that lie between $P1$ and $P2$.

The line is drawn on the screen when there are two end points as $P1$ and $P2$, and the computer fills it with pixels by the intermediate point values. This can be done with various line drawing algorithms like DDA and Bresenham's.

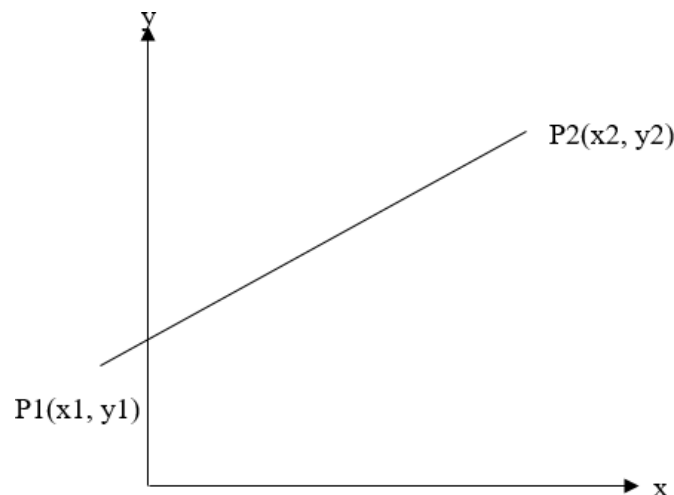


Fig 34. Line representation on coordinate system

DDA Line Drawing Algorithm

DDA stands for Digital Differential Analyzer. This algorithm works on an incremental approach.

Drawing Concept

- The next coordinate to be plotted is calculated based on the

previous coordinates.

- The line to be drawn is based on the analysis of difference between the pixel points.
- The incremental points are calculated based upon the start point and the end point of the line.
- It calculates pixel positions along a line by taking unit step increment along one direction and calculating corresponding coordinate position based on the rate of change of the coordinate (Δx or Δy).
- The slope of the line is the ratio of difference of y coordinates and difference of x coordinates.

$$\Delta y = (y_2 - y_1)$$

$$\Delta x = (x_2 - x_1)$$

where,

(x_1, y_1) is the start point

(x_2, y_2) is the end point

Δy is the difference between y coordinates

Δx is the difference between x coordinates

- The value of the slope will be either positive or negative.
- If the value of the slope is positive, then Δx and Δy are incremented, else decremented.
- For each pixel we move right (along the x-axis), we need to move down (along the y-axis) by m pixels (based on screen coordinates).

Algorithm:

1. Start
2. Input the two end points of a line as (x_1, y_1) and (x_2, y_2)
3. Compute
 $dx = x_2 - x_1$ and $dy = y_2 - y_1$
4. if $\text{abs}(dx) > \text{abs}(dy)$
 $\text{step} = \text{abs}(dx)$
else
 $\text{step} = \text{abs}(dy)$
5. Compute
 $x_{\text{inc}} = dx / \text{step}$
 $y_{\text{inc}} = dy / \text{step}$
6. Initialize $x = x_1$ and $y = y_1$
7. Plot pixel (x, y)
8. For $k=1$ to step
 {

```
x = x + xincy
y = y + yinc
Plot pixel (ROUND(x), ROUND(y));
}
9. End
```

Merits:

- It is simple.
- It is fast to implement.

Demerits:

- It involves floating values, which requires rounding off to plot on screen.
- Rounding to integers takes time.

Example of DDA Line Drawing Algorithm:

- Consider a line PQ with P = (0, 0) and Q = (7, 4). Apply the simpleDDA algorithm and calculate the pixel on this line.

Solution to trace line PQ.

- Start
- Input $x_1=0, y_1=0, x_2=7, y_2=4$
- Compute
$$dx=x_2-x_1=7-0=7$$
$$dy=y_2-y_1=4-0=4$$
- $abs(dx)=7$ and $abs(dy)=4$, so $abs(x)>abs(y)$
so, $step=abs(dx)=7$
- Compute
$$xinc=dx/step=7/7=1$$
$$yinc=dy/step=4/7=0.6$$
- Initialize $x=0, y=0$
- Plot the pixel (0, 0)
- For $k=1$ to 7,
Calculate new values of x and y as $x = x + xinc$ and $y = y + yinc$.
Calculated values shown in Table 1.
- End

Table 1. Pixel value to compute line PQ

k	Plot	x	y
	0, 0	0	0
1	1, 1	1	0.6
2	2, 1	2	1.2
3	3, 2	3	1.8
4	4, 2	4	2.4
5	5, 3	5	3
6	6, 4	6	3.6
7	7, 4	7	4.2

Bresenham's Line Drawing Algorithm:

Bresenham's line drawing algorithm determines the points of an n- dimensional raster that should be selected in order to form a close approximation to a straight line between two points.

This algorithm provides the fast and efficient way to represent continuous line on a discrete plane of a computer display.

The algorithm is suitable for digital plotter and CRT.

Algorithm:

1. Start
2. Input the line end points (x1, y1) and (x2, y2)
3. Plot pixel (x1, y1)
4. Compute
$$dx = x2 - x1$$
$$dy = y2 - y1$$
5. Initialize $p_k = 2*dy - dx$
6. For each x along the line, starting at $k = 0$, perform the following test
If $p_k < 0$,
 Plot pixel (x_k+1, y_k)
 $p_{k+1} = p_k + 2*dy$
Else
 Plot pixel (x_k+1, y_k+1) $p_{k+1} =$
 $p_k + 2*dy - 2*dx$
7. Repeat step 6, $dx - 1$ times.
8. Plot (x2, y2)
9. End

Merits:

- It generates only integer values, so simple to use.
- It is faster than DDA line drawing algorithm.
- It avoids generation of duplicate points.
- Points generated are more accurate.

Demerits:

- The resultant line obtained is not smooth.
- This algorithm is for basic line drawing. It cannot handle zigzag lines.

Example of Bresenham's Line Drawing Algorithm:

- Consider a line AB with A = (0, 0) and B = (7, 4). Apply the Bresenham's line algorithm and calculate the pixel on this line.
- Solution to trace line AB.
 1. Start
 2. Input x1=0, y1=0, x2=7, y=43.
Plot (0, 0)
 4. Compute
$$dx = x2 - x1 = 7 - 0 = 7$$

$$dy = y_2 - y_1 = 4 - 0 = 4$$

5. Initialize $p_k = 2 \cdot dy - dx$,

$$p_k = 2 \cdot 4 - 7 = 8 - 7 = 1$$

6. For $k=0$ to 6,

Calculate new values of x and y based upon decision parameter p_k

If $p_k < 0$,

increment x , y will be same, plot (x_k+1, y_k)

compute $p_{k+1} = p_k + 2 \cdot dy$

else

increment x and y , plot (x_k+1, y_k+1)

compute $p_{k+1} = p_k + 2 \cdot dy - 2 \cdot dx$

Calculated values shown in Table 2.

7. Repeat the steps $dx-1=7-1=6$ times

8. Plot (7, 4)

9. End

Table 2. Pixel value to compute line AB

k	p_k	p_{k+1}	Plot	x	y
			0, 0	0	0
0	1	-5	1, 1	1	0.6
1	-5	3	2, 1	2	1.2
2	3	-3	3, 2	3	1.8
3	-3	5	4, 2	4	2.4
4	5	-1	5, 3	5	3
5	-1	7	6, 4	6	3.6
6	7	1	7, 4	7	4.2

SCAN CONVERSION OF CIRCLE

A circle is defined as a set of points that are at the same distance from center and the distance from center to any point on the circumference of the circle is called a radius.

- It is an eight-way symmetry as in Fig 6, which can be divided into four quadrants, and each quadrant into two octants.
- This symmetry helps in implementation of any circle-drawing algorithm.
- The symmetric property of the circle is utilized to scan convert the circle by plotting eight points for eight octants.

- This means for every point, eight points can be plotted as (x, y) , (y, x) , $(-y, x)$, $(-x, y)$, $(-x, -y)$, $(-y, -x)$, $(y, -x)$ and $(x, -y)$ as shown in Fig 6.
- The entire circle can be obtained by just computing the first octant and then repeat the reflection of the point about each 45° axis.
- It produces seven more points on the circumference of the circle. Hence, it is sufficient to calculate only one octant.
- The circle centered at the origin can be described by the equation: $x^2 + y^2 = r^2$
where,
x and y are the coordinates
r is the radius
- The second approach describe the circle using trigonometric relation, where x and y coordinates of the circle is given by $x = r \cos \theta$ and $y = r \sin \theta$, and θ is the current angle.

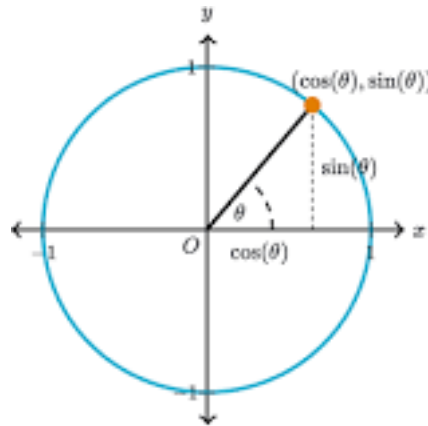


Fig 35. Representing circle in trigonometric relations

Midpoint Circle Algorithm:

This algorithm is used to determine the points needed for rasterizing the circle.

- The perimeter points will be calculated for first octant and will be printed along the mirror points in other octants.
- For any given point (x, y) , the next point to be plotted can be $(x, y+1)$ or $(x-1, y-1)$ based upon the three conditions.
- Given a circle centered at $(0, 0)$, with radius r and point $P(x, y)$, $f(P) = x^2 + y^2 - r^2$
 - If $f(P) < 0$, the given point lies inside the circle boundary, then the upper pixel can be chosen.
 - If $f(P) = 0$, the given point lies on the circle boundary, then any pixel can be chosen.
 - If $f(P) > 0$, the given point lies outside the circle boundary, then the lower pixel is chosen.

Algorithm:

1. Start
2. Declare x , y , r , P , x_c , y_c where (x_c, y_c) are center coordinates, x and y points to be plotted, r is the radius, P is decision factor.
3. Initialize $x = 0$ and $y = r$.
4. Compute decision factor $P = (5/4) - r$.
5. Repeat the steps while $x \leq y$
6. Plot (x, y)
7. If $P < 0$
 Set $P = P + 2x + 3$
Else if $P \geq 0$
 Set $P = P + 2(x - y) + 5$
 $y = y - 1$
8. do $x = x + 1$
9. End

Merits:

- It is an efficient algorithm.
- It is easy to implement.
- It is based on simple circle equation $x^2 + y^2 = r^2$
- It helps to create curves on raster display.

Demerits:

- It is a time consuming algorithm.
- Sometimes the point of the circle are not accurate.

Bresenham's Circle Algorithm :

- The Bresenham's circle drawing algorithm calculates all the points nearest to the circle boundary.
- It is an incremental method where one coordinate of the point is incremented and another coordinate is calculated according to it.
- It uses integer arithmetic, which makes it less complex and working is faster.
- The strategy is to select the pixel, which is at the least distance from the true circle boundary, and then keep calculating the successive points on the circle.
- The circle follows the symmetric property.

- In this algorithm, at any point (x, y) we have two options either to choose the next pixel at the top i.e. (x+1, y) or at the bottom i.e. (x+1, y-1).
- This is done using decision parameter d where, If $d > 0$, (x+1, y-1) will be the next pixel to plot Else (x+1, y) will be the next pixel
- To draw a circle with center (x_c, y_c) and radius r, it will start from (0, r) and move in first quadrant till $x=y$ with initial conditions:

$d = 3 - 2 * r$ $x = 0$

and $y = r$

Algorithm:

1. Start
2. Declare x, y, r, D, x_c , y_c where (x_c, y_c) are center coordinates, x and y points to be plotted, r is the radius, D is decision factor.
3. Calculate decision parameter D as: $D = 3 - (2 * r)$
4. Initialize $x = 0$, $y = r$
5. Compute next pixels on circle based upon decision parameter

While $x \leq y$
 Plot (x, y) if
 $D < 0$, then

$D = D + 4x + 6$

 else

$D = D + 4(x - y) + 10$

$y = y - 1$

 end if

$$x = x + 1$$

6. End

Merits:

- It is a simple algorithm.
- It is easy to implement.
- It is based on simple circle equation $x^2 + y^2 = r^2$

Demerits:

- It is not suitable for high graphics and complex images.
- Accuracy issues while generating points.

Midpoint Ellipse Algorithm:

Ellipse is defined as the geometric figure, which is set of all points on a plane whose distance from two fixed points known as foci remains constant.

- It has two axis, namely major axis and minor axis, where the major axis is the longest diameter and minor axis is the shortest diameter.
- Ellipse has a four-way symmetry property, which means quadrants are symmetric and not the octants.
- Midpoint ellipse algorithm plots the point of an ellipse on the first quadrant by dividing the quadrant into two regions.
- Each point (x, y) is projected to other three quadrants $(-x, y)$, $(x, -y)$ and $(-x, -y)$ as it uses four-way symmetry.

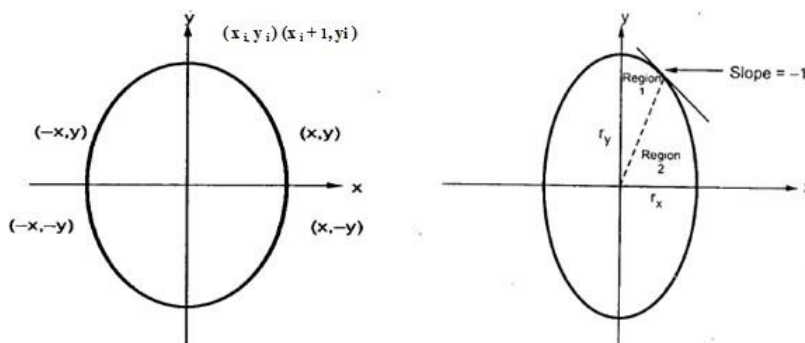


Fig 36. Four-way symmetry and Criteria for Ellipse

- Consider the equation of ellipse as $f(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$ where,
 r_x is the x radius
 r_y is the y radius
- For any point (x, y) the next point to be plotted is based on the three conditions as:

- If $f(x, y) < 0$, then (x, y) is inside the ellipse
- If $f(x, y) > 0$, then (x, y) is outside the ellipse
- If $f(x, y) = 0$, then (x, y) is on the ellipse
- There will be two decision parameters based upon the two regions, P1 for region 1 and P2 for region 2.
- For region 1, $P1 = r_y^2 + r_x^2 / 4 - r_y r_x^2$
- For region 2, $P2 = r_y^2 (x + 1 / 2)^2 + r_x^2 (y - 1)^2 - r_x^2 r_y^2$

ALIASING AND PROBLEMS IN ALIASING

- In computer graphics, smooth curves and other lines become jagged because the resolution of the graphics device or file is not high enough to represent smooth curve.
- In line drawing algorithms, not all rasterized locations match with the true line and we have to select the optimum raster locations to represent a straight line. This problem is more in low-resolution screens.
- In such screens, lines appear like a stair-step. This effect is called aliasing.
- The aliasing effect can be reduced by adjusting the intensities of the pixel along the line.
- This process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called antialiasing.
- The aliasing effect can be minimized by increasing resolution of the raster display device.

WHAT IS CLIPPING?

A procedure that identifies the portions of a picture that are either lying inside or outside of a specified region of space is referred to as clipping and can be done by using the clipping algorithms.

- Applications of clipping include extracting part of a defined scene for viewing, identifying visible surface in three-dimensional views, object boundary, displaying a multi-window environment and many more. The clip window can be a general polygon or any curved boundary.
- For viewing transformation, we want to display only those picture parts that are within the window area.
- Everything outside the window is discarded.
- Clipping algorithm can be applied in world coordinates, so that only the contents of the window interior are mapped to the device

coordinates.

- Clipping can be done on a point, lines, polygons and text.
- In Figure, lines and points within window should be kept (line (P5-P6) and point P8) and the line lying outside the window should be clipped (line (P1 to P2), line (P3 to P4) and point P7).

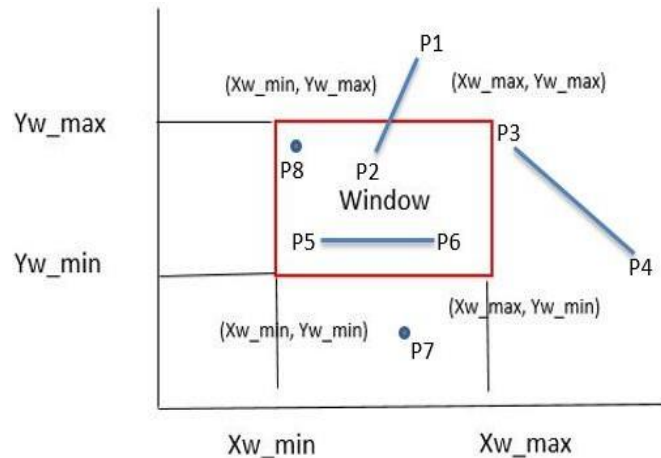


Fig 37. Example of clipping

POINT CLIPPING

Point clipping is a process to define the position of the point.

Point clip

Consider the clipping window is a rectangle in the standard position, a point $P(x, y)$ is used for display with the following constraints

$$xw_{\min} \leq x \leq xw_{\max} \text{ AND } yw_{\min} \leq y \leq yw_{\max}$$

where edges of the clip window are $xw_{\min}$, $xw_{\max}$, $yw_{\min}$ and $yw_{\max}$ then the point is clipped.

- In Figure, point B and point C are within the window range ($xw_{\min}$, $xw_{\max}$, $yw_{\min}$, $yw_{\max}$) so they are not clipped, while point A, point D and point E lie outside the window so they are clipped.

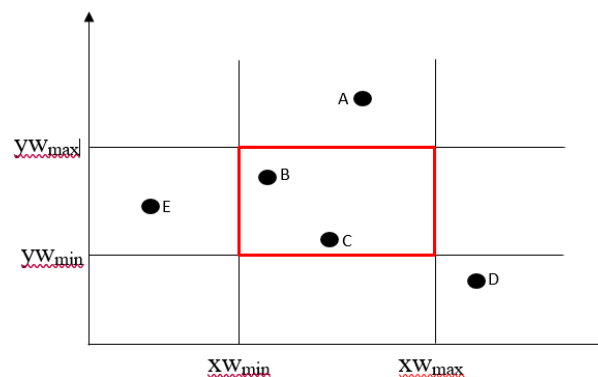


Fig 38. Example of Point Clipping

LINE CLIPPING

- Removing the portion of the lines which are lying outside the edges of the window is called as line clipping.

Procedure for Line Clipping.

1. Test a given line segment to determine whether it lies completely inside the clipping window,
2. If it does not, we try to determine whether it lies completely outside the window.
3. If the line segment is not completely inside or completely outside, perform intersection calculations with one or more clipping boundaries, with the use of various line clipping algorithms.

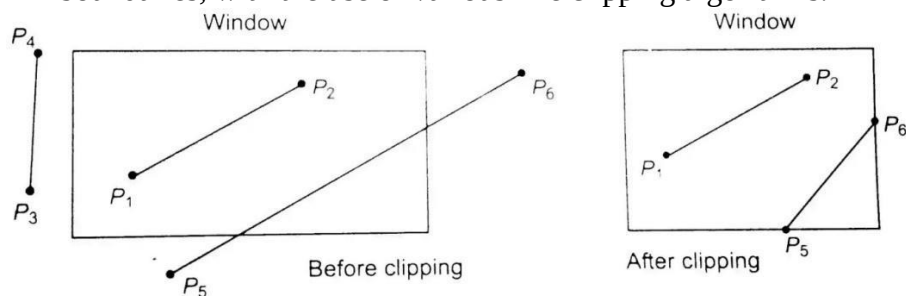


Fig 39. Example of Line Clipping

Examples

- In Figure, line P1 to P2 lies completely inside the window, so it does not require clipping and is saved.
- Line P3 to P4 lies completely outside the window, so it does not require clipping and is not saved.
- Line P5 to P6 lies inside and outside partially, so it requires clipping and is clipped as shown in After Clipping of Fig 4.

Cyrus-Beck Line Clipping Algorithm:

Cyrus Beck Line clipping algorithm is a parametric line-clipping algorithm. The term parametric means it is required to find the value of the parameter „t“ in the parametric representation of the line segment for the point at that the segment intersects the clipping edge.

- In Figure, XY is the line segment, which is intersecting at the two edges A and B of the convex window.
- Apply the parametric equation of the line segment

$$XY \text{ as: } X + t(Y - X)$$

where, $0 \leq t \leq 1$ and t is the linear parameter that continuously changes the value

- Therefore,

$$X + t(Y - X) = (x_1, y_1) + t((x_2 - x_1), (y_2 - y_1))$$

$= (x, y)$ be any point on line XY

- i. While $t = 0$, point X is obtained.
 - ii. While $t = 1$, point Y is obtained.
 - iii. While $0 \leq t \leq 1$, line in between point X and Y is traced.
 - iv. $t = 0.5$, finds the midpoint of line XY .
 - v. While $t < 0$, trace the line on left hand side of X .
 - vi. While $t > 0$, trace the line on right hand side of Y .
- The variation in parameter „ t “ generates line using pointwise method.
 - The range of parameter values will identify the portion to be clipped through any convex polygonal region consisting of n -vertices or lattice points to be identified through the user.

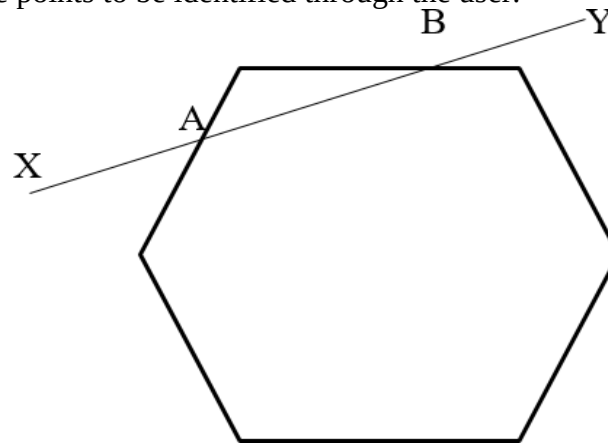


Fig 40. Interaction of line XY with convex polygon window

Merits:

- It follows parametric approach.
- It is fast and efficient.

Demerits:

- The algorithm is can be applicable only for 2D line clipping.

Cohen-Sutherland Line Clipping Algorithm :

Cohen-Sutherland algorithm is a line-clipping algorithm that cuts lines to portions, which are within a rectangular window.

- It removes the portion of the lines from a given set of lines and rectangle area of interest also known as view port, which belongs outside the area of interest and clip those lines that are partially inside the area of interest.

- The algorithm divides a two-dimensional space into 9 regions (one inside and eight outside) and then efficiently determine the lines and portions of lines that are visible in the central region of interest.
- Figure, defines the clipping regions for Cohen-Sutherland algorithm.

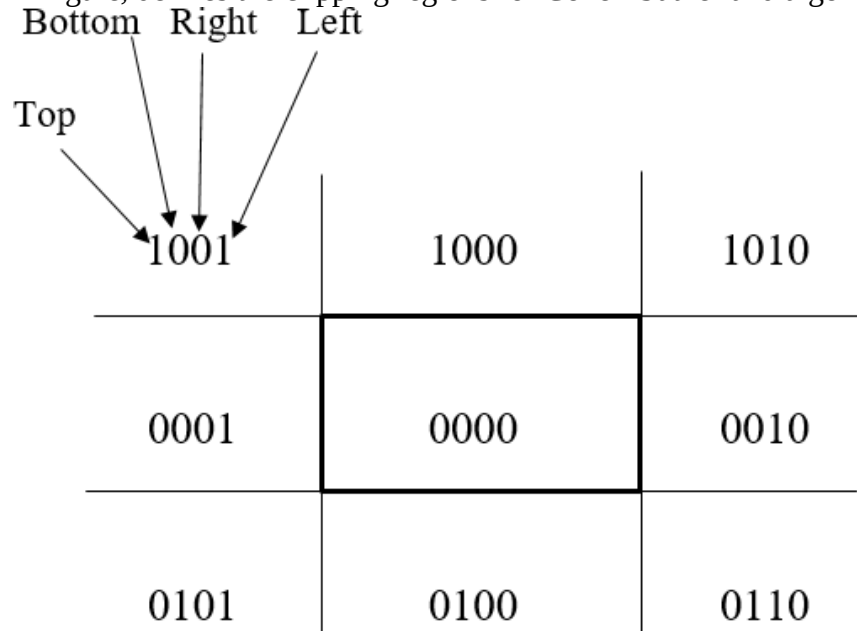


Fig 41. Clipping regions

- It uses 4-bits to divide the entire region, and the centre region is the clipping window.
- These 4 bits represents the TOP, BOTTOM, RIGHT and LEFT of the region.
- In the Figure, the TOP and LEFT bit is set to 1 as it is in the TOP-LEFT corner.
- There are three possibilities for the line:
 - If the line is completely inside the window, the line is accepted.
 - If the line is completely outside the window, the line is rejected completely.
 - If the line is partially inside the window, find the intersection point and only draw that portion of the line that is inside the region.

Algorithm:

1. Assign a region code for each endpoint.
2. If both endpoints have a region code 0000 then accept this line.
3. Else perform the logical AND operation for both region codes
 - 3.1 If the result is not 0000, then reject the line.

- 3.2 Else clipping is required.
 - 3.2.1 Choose an endpoint of the line that is outside the window.
 - 3.2.2 Find the intersection point at the window boundary based on the region code.
 - 3.2.3 Replace the endpoint with the intersection point and update the region code.
 - 3.2.4 Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.
4. Repeat step 1 for other lines.

Merits:

- This algorithm follows encoding approach.
- Easy to use

Demerits:

- This algorithm is applicable to rectangular window only and not to the other convex shaped window.
- Less efficient.

Liang-Barsky Line Clipping Algorithm :

This algorithm is more efficient than Cohen-Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping.

- It is a faster parametric line-clipping algorithm.
- It is based upon two concepts during clipping:
 - It is based upon the parametric equation of the line.
 - The intersections between the line and the clip window are determined by the inequalities describing the range of the clipping window.
 - The parametric equation is given

$$x = x_1 + t * dx$$

$$y = y_1 + t * dy$$
 where, $dx = x_2 - x_1$, $dy = y_2 - y_1$ and $0 \leq t \leq 1$
 - Liang-Barsky line-clipping algorithm uses four inequalities with two parameters p and q, as defined in the algorithm below.

Algorithm:

1. Start

2. Read the two endpoints of the line as $p_1 (x_1, y_1)$ and $p_2 (x_2, y_2)$.
3. Read the two corners of the clipping window as xw_{min} , yw_{min} , xw_{max} , yw_{max} .
4. Calculate the values of the parameters p_i and q_i for $i = 1$ to 4 such that:

$$\begin{aligned} p_1 &= -dx, & q_1 &= x_1 - xw_{min} \\ p_2 &= dx, & q_2 &= xw_{max} - x_1 \\ p_3 &= -dy, & q_3 &= y_1 - yw_{min} \\ p_4 &= dy, & q_4 &= yw_{max} - y_1 \end{aligned}$$
5. If $p_i = 0$, then the line is parallel to i^{th} boundary.
 If $q_i < 0$, then:

The line is completely outside the boundary so discard the line.

Else:

Check whether line is horizontal or vertical and then check the line endpoints with the corresponding boundaries.

If the line endpoints lie within the bounded area then: Use them to draw the line,

Else :

Use boundary coordinates to draw the line.

6. Initialize $t_1 = 0$ and $t_2 = 1$
7. Calculate q_i / p_i for $i = 1$ to 4
8. Select values of q_i / p_i where $p_i < 0$ and assign maximum out of them as t_1 .
9. Select values of q_i / p_i where $p_i > 0$ and assign minimum out of them as t_2 .
10. If ($t_1 < t_2$)

{

$$\begin{aligned} newx1 &= x_1 + t_1 * dx \\ newx2 &= x_1 + t_2 * dx \\ newy1 &= y_1 + t_1 * dy \\ newy2 &= y_1 + t_2 * dy \end{aligned}$$
 Draw line ($newx1, newx2, newy1, newy2$)
 }
11. End

Merits:

- It uses the parametric approach.
- It is more efficient than any other line-clipping algorithms.

POLYGON CLIPPING

A shape with set of connected lines is a polygon.

- Polygons are clipped based on the window, the portion inside the window is kept as it is and portion outside the window is clipped.
- It clips the four edges in the boundary of the clip rectangle.
- The clip boundary determines the visible and invisible regions of polygon clipping and it is categorized in four ways:
 - If visible region is fully inside the clip window, endpoints are saved.
 - If visible region is fully outside the clip window, no endpoints are saved.
 - If visible exits the clip window, save the intersection.
 - If the visible enters the clip window, save the endpoint and the intersection.
 - Each edge of the polygon must be tested against each edge of the clip rectangle, new edges must be added and existing edges must be discarded, retained or divided.

There are two types of polygons: (i) Concave (ii) Convex.

(i) Concave polygon:

It does not have any part of its diagonal in its exterior and at least one angle should be greater than 180° .

(ii) Convex polygon:

It has at least one part of diagonal in its exterior and all angles should be less than 180° .

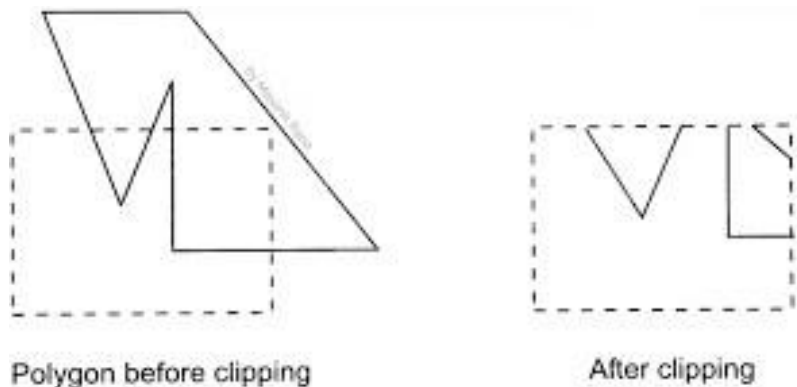


Fig 42. Polygon Clipping

Sutherland-Hodgeman Polygon Clipping :

Sutherland-Hodgeman polygon-clipping algorithm uses divide and conquer strategy.

Procedure:

- It first clips the polygon against the right clipping boundary.
- The resulting polygon is then clipped against the top boundary.
- The same process is repeated for remaining two boundaries.
- It works in order and makes sure that all edges of clipping polygon are taken sequentially.
- At each step new sequence of vertices are generated and passed to next window boundary for clipping.
- To clip the vertices following cases are considered:

Case 1- Both vertices are inside:

- Only the second vertex is added to the output list.

Case 2 - First vertex is outside while second one is inside:

- Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list.

Case 3 - First vertex is inside while second one is outside:

- Only the point of intersection of the edge with the clip boundary is added to the output list.

Case 4 - Both vertices are outside:

- No vertices are added to the output list.



Fig 43. Before Clipping

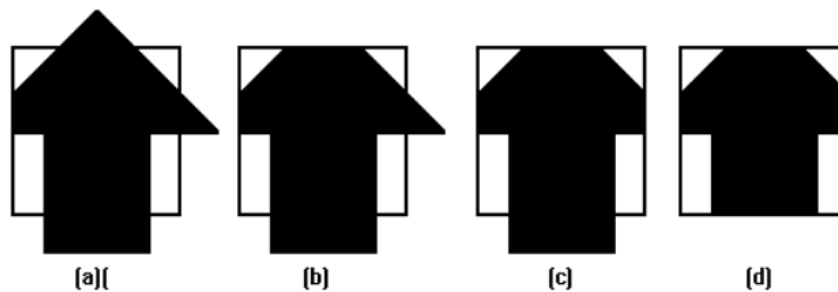


Fig 44. After clipping using Sutherland-Hodgeman algorithm

Example:

- a. Clipping against the left side of the clip window.
- b. Clipping against the top side of the clip window.
- c. Clipping against the right side of the clip window.
- d. Clipping against the bottom side of the clip window.

Sutherland-Hodgeman comprises of two key processes in the algorithm:

- Perform inside-outside test to determine the visibility of a point or vertex.
- Determine the intersection of the polygon edge and the clipping plane. Method to determine the visibility of a vertex or a point.
- Let V be the vertex and AB the window boundary.
- Three points A, B and V define Vector AB and AV line in the plane.
- If this plane is considered in the xy plane, then the vector cross product $AV \times AB$ has a z component given by:

$$(x_v - x_a)(y_b - y_a) - (y_v - y_a)(x_b - x_a)$$

- The sign of the z-component decides the position of point V with respect to window boundary.
- It can be as follows:
 - $z > 0$: point is on the right side of window boundary
 - $z = 0$: point lies on the window edge
 - $z < 0$: point is on the left side of window boundary

Algorithm:

1. Read the coordinate of vertices of the subject polygon and clipping polygon.
2. Consider an edge of the clipping window, compare the vertices of each edge of the subject polygon with the clipping polygon or the edge, and record the intersections.
3. Store the new intersection and vertices in the new list of the vertices as per the cases.
4. Perform steps 2 and 3 for the remaining edges of the clipping polygon

each time, the resulting list of polygon vertices are successively passed to process the next edge of the clipping polygon.

5. Finish.

Merits:

- The process is simple and efficient.

Demerits:

This method requires a considerable amount of memory.

TRANSFORMATIONS AND MATRICES

Transformation:

Creation of graphs and images in computer are not sufficient in computer graphics. In real world object are moved from one position to another or its changing its shape or orientation and it can be seen from different angles and different perspective.

The process of changing size, shape, position and orientation of object is called transformation.

Matrices:

- It provides the technique to represent the real world object into computer system.
- It also provides a way to calculate different type of geometric function on the real life object to show on the screen.
- With help of matrices various operation like translation scaling etc can be done on the object

TRANSFORMATION CONVENTIONS

To draw an object onto screen, computer need the system that is called as conventions of transformation. This can be done with 2 types such as 2D coordinate system and 3D coordinate system.

For 2D image we use 2D coordinate system as convention for transformation whereas for 3D image we use 3D coordinate system as convention for transformation of image.

2D COORDINATE SYSTEM:

This 2D coordinate system consists of two coordinates x and y i.e. x plane and y plane. Its direction is along with the x and y axis with reference to an origin.

With the help of this system the real word object can be draw onscreen with lines, circle, polygon etc

3D coordinate system:

This 3D coordinate system consists of three coordinates x, y and z .i.e. x plane, y plane and z plane.

Its direction is along with the x, y and z axis with reference to an origin. With the help of this system the real word object can be drawn on screen with lines, circle, polygon etc.

2D TRANSFORMATIONS

The transformation which is applied on 2 plane i.e. x and y plane to change the appearance of the object is called 2D transformation.

Basic 2D transformation	1. Translation	Denoted by T
	2. Rotation	Denoted by R
	3. Scaling	Denoted by S
	4. Shearing	Denoted by Sh
	5. Reflection	Denoted by RE

Translation:

- Changing the position of an object or shifting of an object is called 2Dtranslation.
- To achieve translation an object should shift in x and y direction.
- We can translate a two dimensional figure by adding **translation factor t_x and t_y** to the original points x and y

Rotation:

Changing the position of an object with rotation angle along with xyplane is called 2D rotation.

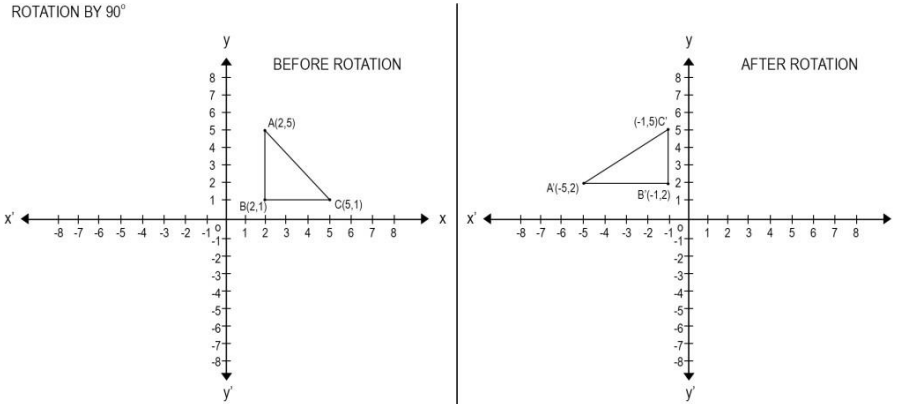


Fig 45. Rotation

Shearing:

- A shear is something that pushes the object sideways and make a thing little bit tilted.
- To achieve shearing an object should tilted(slant) with respect to x and y direction.
- We can shear a two dimensional figure by multiplying **shearing factor sh_x and sh_y** to the original points x and y

Reflection:

- Creating mirror image of an object is called 2D reflection.
- To achieve reflection an object should appear in the form of mirror object in x and y direction.
- We can reflect a two dimensional figure by multiplying **reflection factor RE_x and RE_y** to the original points x and y

COMBINED TRANSFORMATION

Sometime an object may require more than one transformation to get desired output, we need to transform an object at several times with different transformation like rotation, scaling, translation, shearing, reflection such a group of transformation is called combined transformation.

The sequence of transformation is applied on an object to get the required shape of object.

It is also called composite transformation.

Method:

When we need to apply more than one transformation on an object first we need to do single transformation in matrix form then we need to apply second transformation on an object in matrix form then the final matrix multiplication is considered for the desired object transformation.

**HOMOGENEOUS COORDINATES AND MATRIX
REPRESENTATION OF 2D TRANSFORMATIONS**

TRANSFORMATION OF POINTS

All types of transformation such as translation, rotation, scaling, reflection, shearing is applied to get the transformation of point. All the transformation elements can be described through matrices to perform transformation on points.

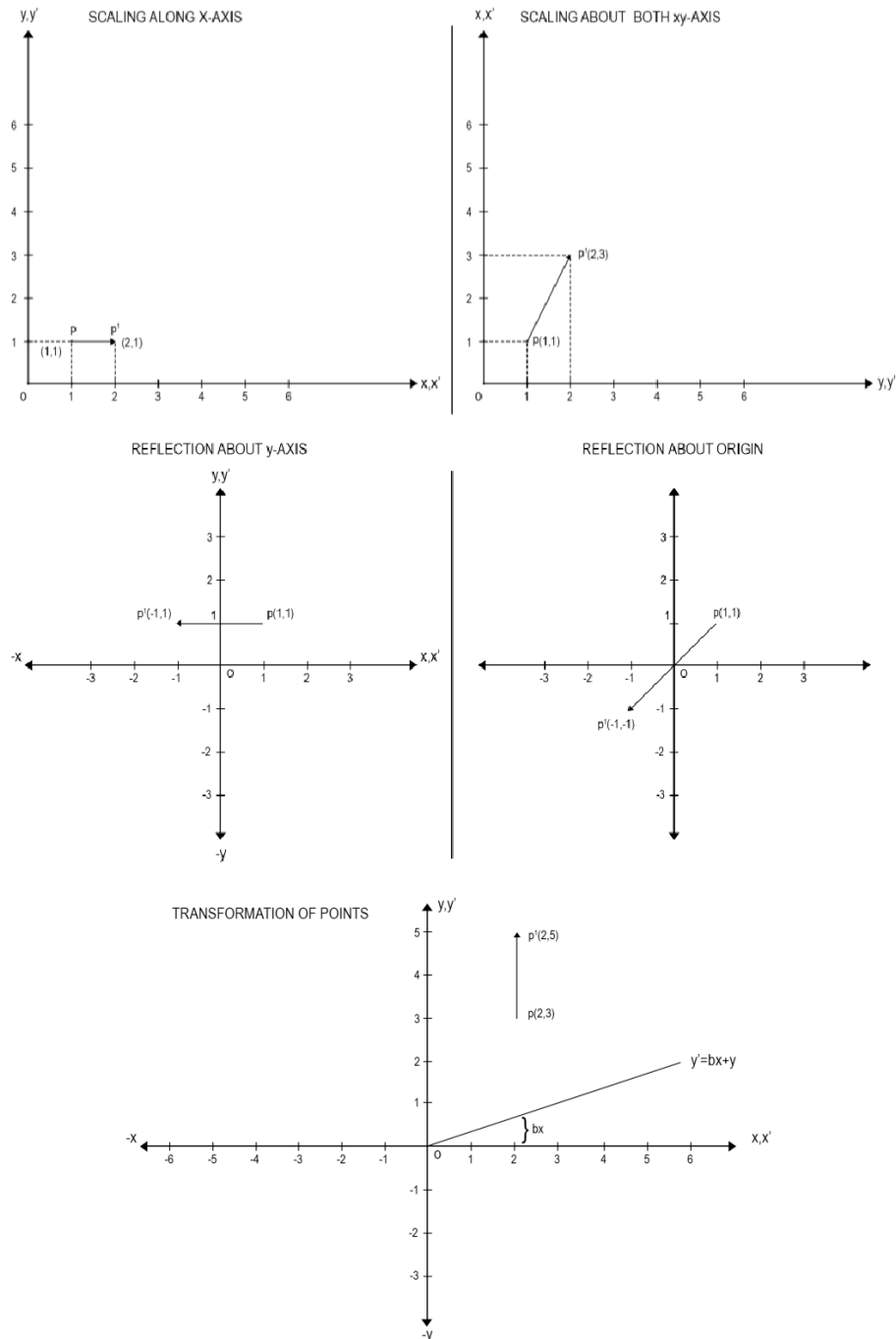


Fig 46 Transformation of point

TRANSFORMATION OF THE UNIT SQUARE

When we transform an object in 2D transformation, all the points are transformed except the origin point it will remain unchanged, which indicates that the conversion of an object in to new shape in 2D plane with xy coordinates. These points are considered as apposition vectors of a unit square with one point situated at the origin of the coordinate system. are same but the rest of the coordinates are changes according to the unit vector , hence the transformation is applied on all point except origin point.

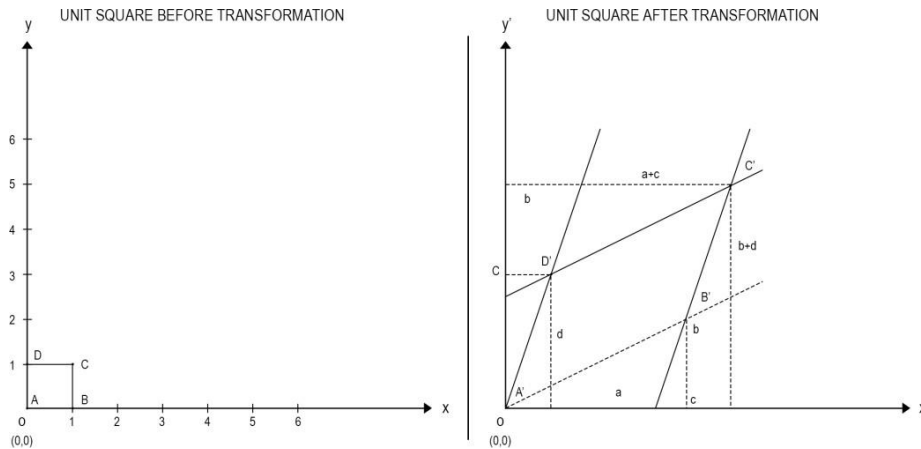


Fig 47 Unit square transformation

SOLID BODY TRANSFORMATIONS

- Solid body transformation is based on vector transformation. Before performing transformation basic physics quantities it should be known.
- We know that in solid body transformation parallel lines are transformed as parallel lines but it is not necessary that perpendicular line will always transformed perpendicular only.
- Angle between lines are also not preserved.
- To preserve an angle between two intersecting line solid body transformation is used. To achieve this, dot product and dot product of vector are required.

Vector: A vector is quantity which has direction as well as magnitude. It is used to determine the position of one point in space to another. Vector is used in graphics also known as vector graphics. Vector graphics is the creation of digital image through sequence of commands of mathematical statements that places lines and shapes in given two dimensional or three dimensional space. Vector graphics describes series of points to be connected.

Dot Product of vector: they can be multiplied using dot product of matrix is written using central dot present between these two vector.

Cross Product of vector: the cross product of two vector is an another vector which is right angled to both the vector like $a \times b$

ROTATION ABOUT AN ARBITRARY POINT

In translation and homogeneous coordinate system we have seen that the homogeneous coordinate system accomplishes the rotation about the points of an origin whereas, 2D transformation does not allow rotation of points at origin.

REFLECTION THROUGH AN ARBITRARY LINE

Reflection is achieved by mirroring the line coordinates on x axis, y axis and even by passing line through origin when reflection is taken place on both x and y axis.

We know that when x and y both the coordinates are changed, the line is reflected and passes through the origin. But sometimes reflection of an object through a line which does not pass through an origin is required.

Step 1: Translation: translate the line and the object so that the line passes through origin. $[T]$

Step 2: Rotation: Rotate the line and the object the origin until the line is coinciding with one of the coordinate axis. $[R]$

Step 3: Reflection: reflect through the coordinate axis $[R]$

Step 4: Inverse rotation: apply inverse rotation about the origin. $[R]^{-1}$

Step 5: Inverse translation: translate back to original location. $[T]^{-1}$

VIEWPORT TRANSFORMATIONS

A world coordinate area selected for display is called a **window**. An area on display device to which a window is mapped is called **viewport**. The window defines what to be display on screen whereas, the viewport defines where is to be displayed. The standard rectangle position is applied to see window and viewport. The part of real world image (world coordinate system) seen on device coordinate system (screen coordinate system) is called as **viewing transformation**. Sometimes two dimensional transformations is also called as **window to viewport transformation**.

In computer graphics, the term window is referred to an area of image that is to be display on the screen (View plane). Whenever we want to display

image on screen the viewing coordinate is rectangular window which shows the enlarge image in viewport. By changing the position of the viewport, we can view object at different position on the display area of output device. Also, by varying the size of viewports, we can change the size and proportions of displayed objects. When all coordinate transformations are completed, viewport clipping can be performed in normalized coordinates or in device coordinates. Viewport is typically defines in unit square.

Process of window to viewport coordinate system:

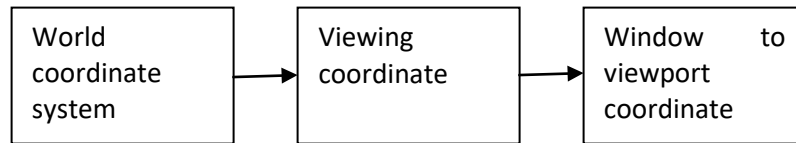


Fig 48 window to viewport system

Window to viewport mapping:

A point at position (x_w, y_w) in the window is mapped into new position (x_v, y_v) in associated view port.

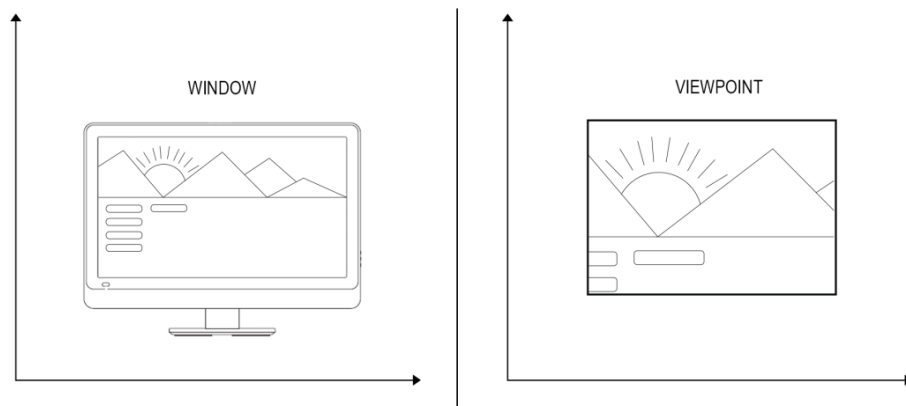


Fig 49 Window to Viewport transformation

Module III: 3D Graphics
3.1 3D Cartesian and Homogeneous co-ordinate system,
3.2 Geometric transformations (Translation, Scaling, Rotation, Reflection),
3.3 Composite transformations.
3.4 Mathematics of Projections: Perspective Projections - Mathematical Description and Anomalies of perspective projections.
3.5 Parallel Projections – Taxonomy of Parallel Projections and their Mathematical Description.
3.6 Introduction to 3D viewing pipeline and 3D clipping.

INTRODUCTION

When we want to convert the real life object into computer we need to understand the coordinate system. The real life objects represent in computer with the different angles and different perspective with the help of different coordinate system and here the concept of transformation exist. Transformation takes place in x axis, y axis, z axis in 3D transformation. In this chapter we will see the 3D coordinate system of converting real life object to computer. The transformation which takes place on 3D plane or 3D space is called as 3D transformations.

TRANSFORMATIONS AND MATRICES

Transformation:

Creation of graphs and images in computer are not sufficient in computer graphics. In real world objects are moved from one position to another or its changing its shape or orientation and it can be seen from different angles and different perspective.

The process of changing size, shape, position and orientation of object is called transformation.

This change can be done by **Geometric transformation**:

In this type of transformation the object changes its appearance with the help of Geometric properties of the specified points.

1. **Coordinate transformation:**
In this type of transformation the object y' appears with the modified points on the screen after performing some non-singular x' operation like translation, rotation, scaling etc.

Matrices:

- In computer graphics, objects can be described in various quantitative numbers also called points. To represent these quantities, matrices are used.
- It provides the technique to represent the real world object into computer system.
- It also provides a way to calculate different type of geometric function on the real life object to show on the screen.

- With help of matrices various operation like translation scaling etc can be done on the object

- General Matrix representation of 3×3 matrix

$$\begin{bmatrix} 1 & 4 & 3 \\ 5 & 9 & 8 \\ 8 & 2 & 4 \end{bmatrix}$$
- Homogeneous Matrix representation of 3×3 matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

TRANSFORMATION CONVENTIONS

To draw an object onto screen, computer need the system that is called as conventions of transformation. This can be done with 2 types such as 2D coordinate system and 3D coordinate system.

For 2D image we use 2D coordinate system as convention for transformation whereas for 3D image we use 3D coordinate system as convention for transformation of image.

3D coordinate system:

- This system is made up of three coordinates: x, y, and z, which correspond to the x, y, and z planes, respectively and its direction along with the x, y and z axis with reference to an origin. With the help of this system, the real word object can be drawn on screen with lines, circles, polygons, etc. Rules applied for transformation

Right hand coordinate system:

- In this coordinate system right hand figures (X-Thumb, Y-index and X-middle) used as axis to represent the position of the transformed image.
- In Right hand coordinate system all coordinates are positive so the object will transform x in positive direction only.

Left hand coordinate system:

- In this coordinate system right hand figures(X-Thumb, Y-index and Z-middle) used as axis to represent the position of the transformed image.
- In Right hand coordinate system x and y coordinates are positive and z coordinate is negative so the object will transform in both the direction.

3D TRANSFORMATIONS

The transformation which is applied on 3 plane i.e. x, y and yz plane to change the appearance of the object is called 3D transformation.

Basic 3D transformation	1.	Translation	Denoted by T
	2.	Rotation	Denoted by R
	3.	Scaling	Denoted by S
	4.	Shearing	Denoted by Sh
	5.	Reflection	Denoted by RE

Translation

- Changing the position of an object or shifting of an object is called 3D translation.
- To achieve translation, an object should shift in the x, y, and z direction.
- We can translate a two dimensional figure by adding **translation factor** t_x , t_y and t_z to the original points x, y and z.

Rotation:

- Changing the position of an object with rotation angle along with xyz plane is called 3D rotation.
- To achieve rotation an object should move in x and y direction with the specified angle θ .
- We can rotate a two dimensional figure by multiplying **rotation factor** r_x , r_y and r_z to the original points x, y and z

Scaling:

- Changing the size of an object is called 3D scaling.
- To achieve scaling an object should change the size of object in x, y and z direction.
- We can scale a two dimensional figure by multiplying **scaling factor** s_x , s_y and s_z to the original points x, y and z
- There are two types of scaling
 - 1 Uniform scaling: if all scaling factors are same then it is called as Uniform Scaling
 - 2 Non-uniform scaling: if all scaling factors are different then it is called as Non- Uniform Scaling

Shearing:

- A shear is something that pushes the object sideways and make a thing little bit tilted.
- To achieve shearing an object should tilted(slant) with respect to x, y and z direction.
- We can shear a two dimensional figure by multiplying **shearing factor sh_x , sh_y and sh_z** to the original points x, y and z.

Reflection:

- Creating mirror image of an object is called 3D reflection.
- To achieve reflection, an object should appear in the form of mirror object in x, y and z direction.
- We can reflect a two dimensional figure by multiplying **reflection factor RE_x , RE_y and RE_z** to the original points x, y and z

MULTIPLE TRANSFORMATION

Sometime an object may require more than one transformation to get desired output, we need to transform an object at several times with different transformation like rotation, scaling, translation, and shearing, reflection such a group of transformation is called combined transformation.

Multiple transformation means that applying more than one transformation on single object by using respective transformation matrix. The sequence of transformation is applied on an object to get the required shape of object.

Method:

When we need to apply more than one transformation on an object first we need to do single transformation in matrix form then we need to apply second transformation on an object in matrix form then the final matrix multiplication is considered for the desired object transformation.

HOMOGENEOUS COORDINATES AND MATRIX REPRESENTATION OF 3D TRANSFORMATIONS

In 2D coordinate system homogeneous coordinate can be represented in 3 X 3 matrix whereas, in 3D coordinate system homogeneous coordinate can be represented in 4 X 34 matrix.

Hence instead of only x, y, z in 3D coordinate system we could use x, y, z, h where $h \neq 0$

ROTATION ABOUT AN ARBITRARY AXIS INSPACE

Rotation about an arbitrary axis is implemented using translation and rotation about the coordinate axis. Hence there should be coordinate axis which is

coincident to the arbitrary rotation axis. In 3D we have rotation about an arbitrary point through coordinate axis which is rotate around itself of the object

Steps to achieve Rotation about an arbitrary Axis in space

1. The origin point which is to be translate $p=[a_x, b_y, c_z]$ through all 3 axis

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a_x & -b_y & -c_z & 1 \end{bmatrix}$$

2. Perform specified rotation about an arbitrary axis x and y. Rotation in anticlockwise angle implies positive direction.
3. Apply Inverse rotation on step 2
4. Inverse Translation of step 1 to make an arbitrary axis in a particular direction except towards origin.

REFLECTION THROUGH AN ARBITRARY PLANE

Reflection is achieved by mirroring the line coordinates on x axis, y axis and even by passing line through origin when reflection is taken place on both x and y axis.

We know that when x and y both the coordinates are changed; the line is reflected and passes through the origin. But sometimes reflection of an object through a line which does not pass through an origin is required.

Step 1: Translation: translate the point P which lies on reflection plane. $[T]$

Step 2: Rotation: Rotate the point about x and y axis. $[R]$

Step 3: Reflection: reflect an object through z axis or xy plane with $z=0$ $[R^{-1}]$

Step 4: Inverse rotation: apply inverse rotation about 2 axes which state in step 2. $[R]^{-1}$

Step 5: Inverse translation: perform inverse translation of step 1 to get back to original location. $[T]^{-1}$

AFFINE AND PERSPECTIVE GEOMETRY

- Study of parallel lines in an object is called affine geometry.
- In affine geometry lines of object are always parallel whereas in perspective geometry lines of object are always perpendicular means not parallel.
- Affine transformation can be achieved by combining translation, scaling, rotation and shearing together on object.
- In affine geometry study of connected lines are concerned.
- To convert one object shape into another shape in three dimensions

affine and perspective transformation is used.

- When 3D object is transformed into 2D object it is called as projection.
- In perspective projection the object is projected by center of projection of 3D image on 2d plane. If the center of projection is located at a finite point in 3D space then it is called as perspective projection.
- Another type of projection in which the center of projection has been move to infinite is called parallel projection.
- In parallel projection the information of the origin object remains same and the maintain realistic view of an image.

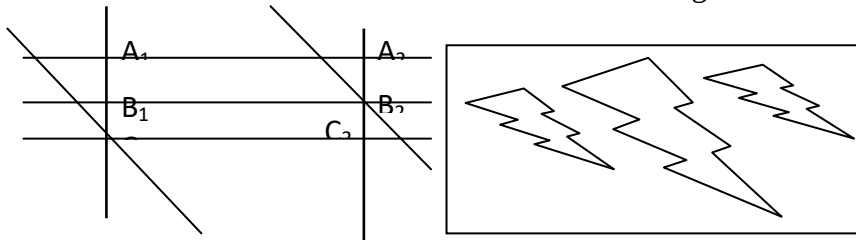


Fig 50 affine transformation and geometry

PERSPECTIVE TRANSFORMATIONS

When human eyes see far away things they are looking smaller as they come closer they are looking bigger to eyes this type of viewing is called perspective. This technique need to convert an object from one form to another and that is known as projection. Perspective projection will convert 3D image into 2D as human eyes does. The projection transforms shape of object into another form as they appear close to eyes.

There are three types of perspective transformation

1. Single point perspective transformation
2. Two point perspective transformation
3. Three point perspective transformation

1. Single point perspective transformation: If one term (either x, y, z) in fourth column of matrix transformation is non zero we will get single point perspective transformation.

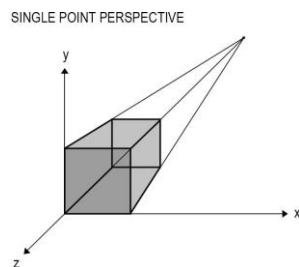


Fig 51 Single point perspective transformation

1. **Two point perspective transformation:** If two term (either x , y , z) in fourth column of matrix transformation is non zero we will get single point perspective transformation. In this type object can be seen in two directions.

TWO POINT PERSPECTIVE

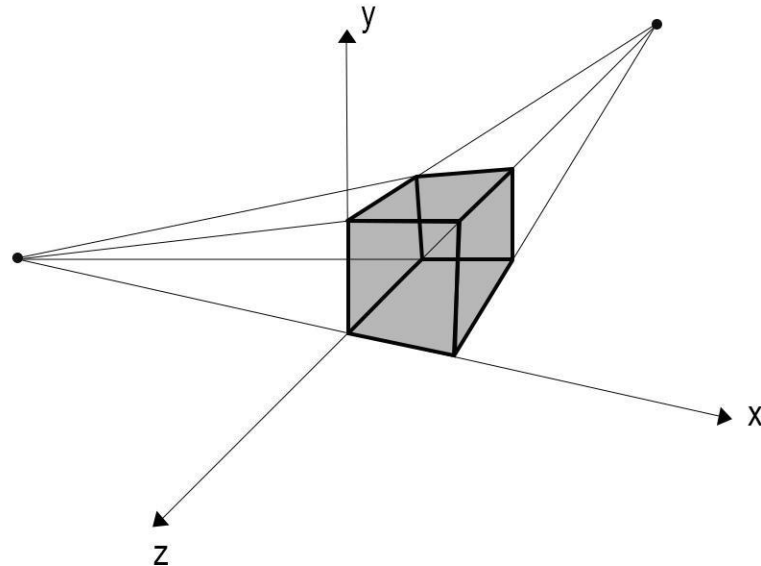


Fig 52 Two point perspective transformation

2. **Three point perspective transformation:** If three term (either x , y , z) in fourth column of matrix transformation is non zero we will get single point perspective transformation. In this type object can be seen in three directions.

THREE POINT PERSPECTIVE

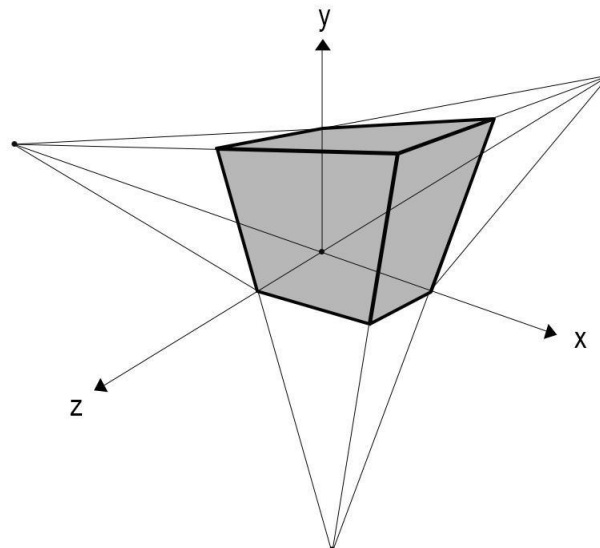


Fig 53 Three point perspective transformation

TECHNIQUES FOR GENERATING PERSPECTIVE VIEWS

Translation and rotation operations are used to create perspective view of an object. When we see object from multiple directions, the object makes 3D view. For a fixed center of projection, a single point perspective projection is carried out by multiple faces (One, two and three perspective). Ref fig 6.10.1, 6.10.2, 6.10.3

1. **Translation of object**
2. **Rotation**

AXONOMETRIC PROJECTION

The problem of rendering for two-dimensional view of dimensional object existed long before the computer was used. One class of projection often used in drafting is called an **axonometric projection**. This is a parallel projection for which the direction of projection is perpendicular to the view plane. We can alter the direction of an **axonometric projection** to get different view of an object, provided we also change the view plane normal to match the projection direction. If we change our direction slightly to the side, then one of the side faces will become visible, while the edges of the front face will shorten. If we raise our angle of view top edges lengthen make the top face visible, while the edges on the side faces appear to shrink.

Types of axonometric projection are

1. **Isometric projection**
2. **Dimetric projection**
3. **Trimetric Projection**

1. Isometric projection: It is mainly used in engineering drawing. In this type of projection the object is viewing from all the direction is looking same and the principal axis are equally distributed on same distance with common angle of 120° among the entire axis. Because of the equal angle among all three axis the distance and the length of the all side of object will be same. All The parallel lines are equal in length.

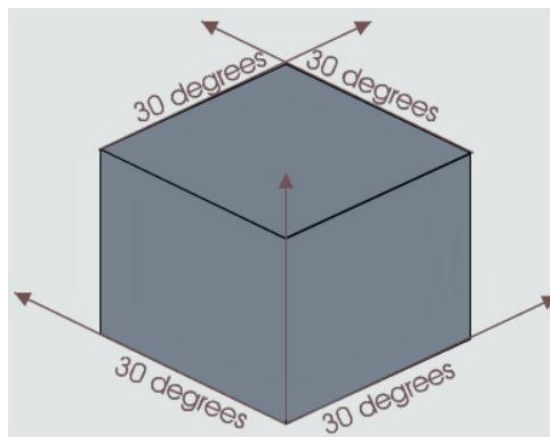
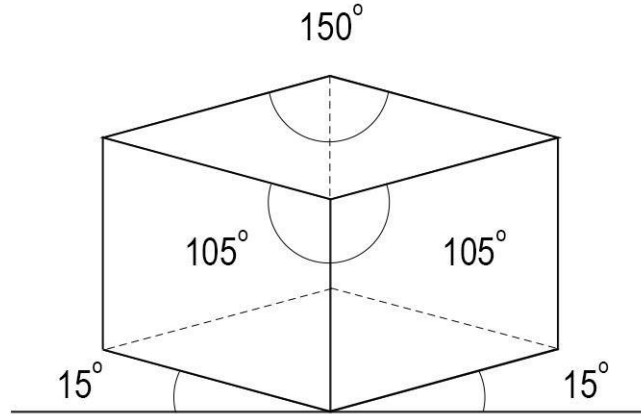


Fig 54 Isometric Projection

2. Dimetric projection: In isometric projection all three axis appears to be equally distanced but in Dimetric projection instead of all three, any two axis are equally distanced and one axis is different distanced than other two. This projection says that the view created through principal axis are represented the angle of object. Since two axis are same the line which are drawn along axis are also same.



DIMETRIC PROJECTION

Fig 55 Dimetric Projection

3. Trimetric Projection: isometric projection all three axis appears to be not equally distanced. All three axis are unequally distanced as well as the angles created by these lines are also unequal.

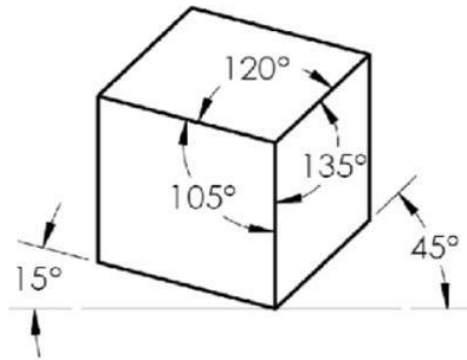


Fig 56 Trimetric Projection

OBLIQUE PROJECTION

If the direction of parallel projection is not parallel to view plane then we have oblique projection. There are two types of oblique projection cavalier projection and cabinet production.

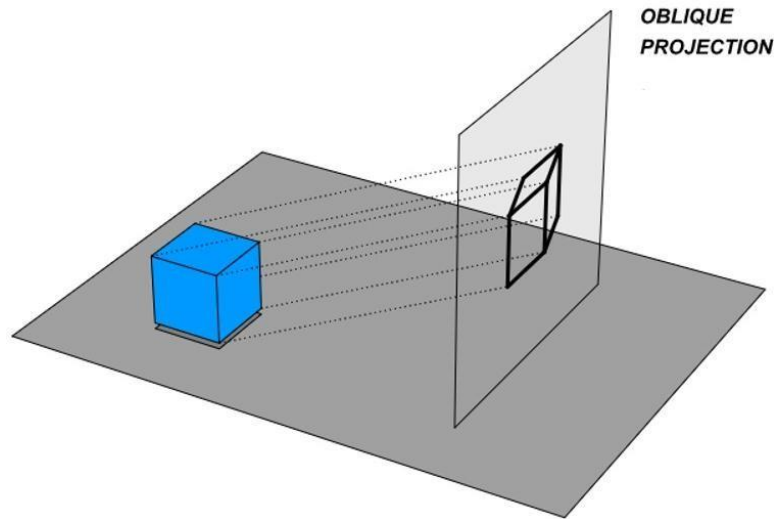


Fig 57 Oblique projection

1. Cavalier projection: Whenever the objects' edge parallel to the co-ordinate axis also will be parallel to the front face that is called as Cavalier projection in this direction of projection is slanted so the points with Z coordinates are positively project down and to the left on the view plane. Z co-ordinate negatively projected up and right face. The angle of z axis can be any of the direction but the distance of the points shifted in a z direction must be equal to three dimensional view plane z distances that is called as Cavalier projection.

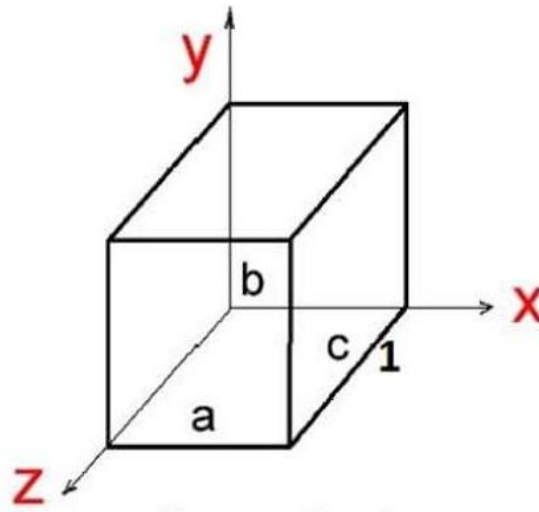


Fig 58 Cavalier Projection

2. Cabinet projection: In cabinet projection, the lines which are perpendicular to the projection plane are projected at one and half of the actual length of the object. The projection is said to be cabinet when an angle between projection plane and oblique projection is equals to 63.4° .

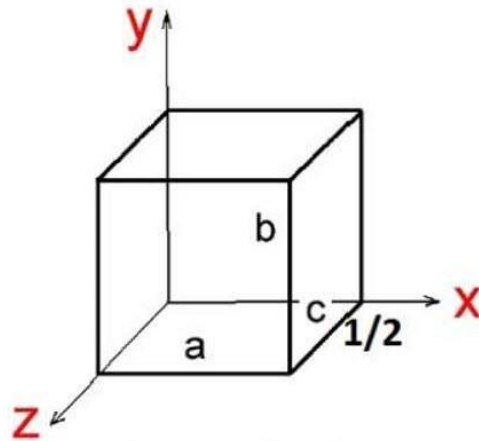


Fig 59 Cabinet Projection

ORTHOGRAPHIC PROJECTION

An orthographic projection is a type of parallel projection in which the projectors are parallel to each other and perpendicular to the plane of projection. It represents 3 dimensional objects in 2 dimensions. A square or rectangular viewing volume is formed for this projection by near, far, right, left, top and bottom clipping planes. Single orthographic projection does not provide sufficient information to reconstruct the shape of an image, therefore, multiple orthographic projection displays a single face of 3D object i.e. viewer can only see the one side of an object at a time.

In multiple projections front surface is parallel to plane of projection. Projection line of sights is perpendicular to the projection plane. Projectors are parallel to each other and originate from any point on object.

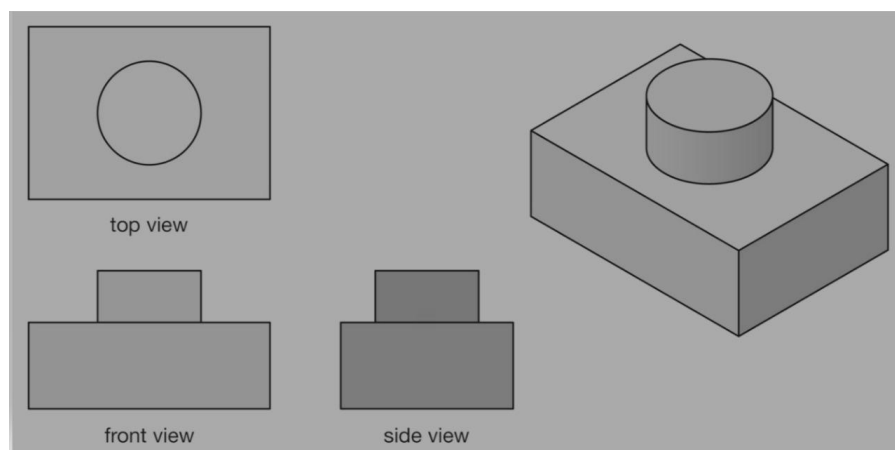


Fig 60 Orthographic Projection

VANISHING POINTS

A vanishing point is abstract point on the image plane where 2D projections of a set of parallel lines in 3D space appear to converge or intersect i.e. certain set of parallel lines appear to meet at different point which is known as vanishing point. When the set of parallel lines are perpendicular to a picture plane then the construction is known as one point perspective and their vanishing point corresponds to the “eye point” from which the image should be viewed for correct respective geometry.

Types of Vanishing point

1. One point perspective:

As the figure 6.14.1 shows end of the parallel line of hotels or houses after certain time of period the parallel lines meet each other at some point that point is called as one perspective vanishing point. Vanishing points are formed by the intersection parallel lines to one of the three axis x , y and z . all three parallel points are perpendicular to the same viewplane at single point.

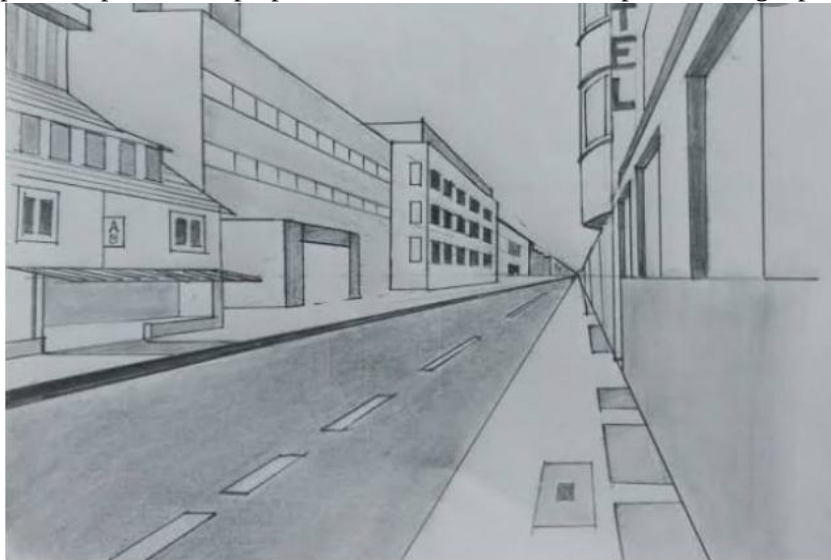


Fig 61 one point perspective vanishing point

2. Two point perspective:

In two points perspective side of an object vanish or recede towards one of the two vanishing points on the horizon.



Fig 62 Two point perspective vanishing point

3. Three point perspective:

In three point perspective, there are three vanishing points where two vanishing points are along the horizon and third one is located either above the horizon or below the horizon.

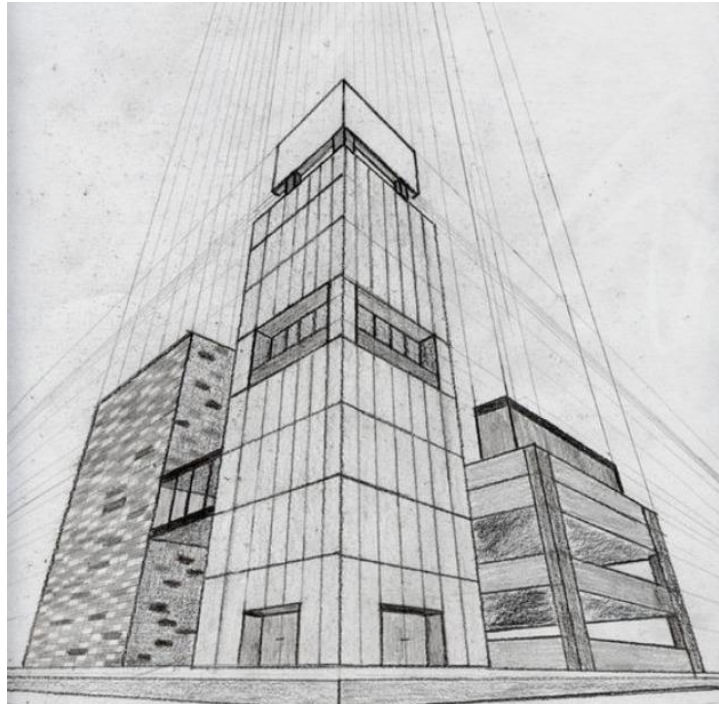


Fig 63 Three point perspective vanishing point

VIEW VOLUMES FOR PROJECTIONS

Converting 3D object into 2D is called projection. Projection operation convert the viewing plane coordinate into 2D coordinate system so that the object will look like same as it seen in the real world. There are two types of projection.

- 1. Parallel projection**
- 2. Perspective projection**

Parallel projection:

In this type of projection parallel line of object transforms in parallel lines only. a parallel projection is also called as relative projection as it preserves relative plane along with the parallel lines. Accurate view of object from multiple sides is view in this projection. This projection will not preserve realistic representation.

Parallel projection classification: 1 Orthographic Parallel projection
2. Oblique projection

Orthographic Parallel Projection: it is done by projecting points along parallel lines that are perpendicular to the projection plane.

Oblique projection: oblique projections are obtained by projecting along with parallel lines that are NOT perpendicular to the projection plane

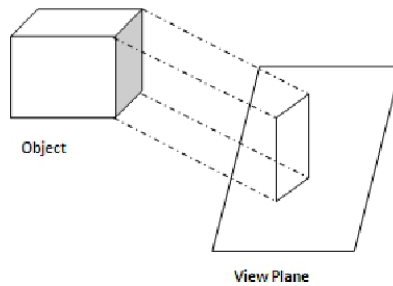


Fig 64 Parallel projection

Perspective projection:

Perspective projection transforms object positions to the view plane while converging to center point of projection. Perspective projection produces realistic views but does not preserve relative proportions. Projections of distant objects are smaller than the projections of object of the same size that are closer to the projection plane.

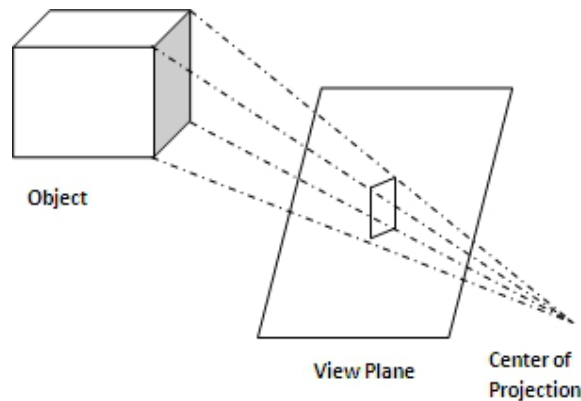


Fig 65 Perspective projection

View Volume:

A view volume is rectangular area in the view plane which controls how much of the scene is viewed. It is formed by the window and the type of projection to be used. View volume is generally defined as a space between front and backplane.

Only those objects are seen which are within the window volume. The objects which are beyond the view volume will not be seen. In view volume, the perspective effects depend on the positioning of the center point of projection. If it is close to the view plane, then the object looks closer and larger, but when it is further from the view plane, it looks smaller and further away. Hence, the size of the object is affected by the relative position of the object. View volume is bounded by 6 planes: left, right, top, bottom, front, and back side.

STAGES IN 3D VIEWING

Viewing in 3D is divided mainly into the following stages:

- Modelling transformation

- Viewing transformation
- Projection transformation
- Normalization transformation and Clipping
- View Port transformation

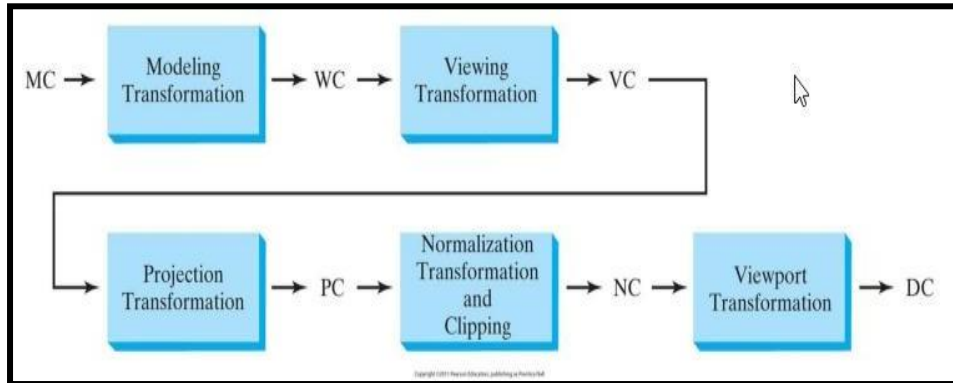


Fig: 66 Stages in 3D Viewing

1. Modelling Transformation:

- Modelling Transformation can be obtained with the help of 3D transformations.
- It transforms the object (Modelling) coordinates (MC) into 3D world coordinate system, will generate output as 3D world coordinates (WC). (3D to 3D).

2. Viewing Transformation:

- The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane.
- It transforms the world coordinate into viewing coordinates (VC) by performing clipping against canonical view volume.

3. Projection Transformation:

- Projection operation converts the viewing-coordinate description (3D) to coordinate positions on the projection plane (2D).
- It projects the coordinates onto projection plane as Projection Coordinates (PC). (3D to 2D).

4. Normalization Transformation and Clipping:

- The coordinates outside the view volume will get clipped and the transformation will get normalized and the Normalized Coordinates (NC) will be generated.

5. Viewport Transformation:

- It maps the coordinate positions on the projection plane of the output device.

- It transforms Normalized Coordinates into viewport is 2D Devicecoordinates (DC) or display.

CANONICAL VIEW VOLUME (CVV)

- Canonical, in computer graphics, is the standard state or behavior of an attribute. This term is used to refer to perceptions that are distinctive and/or natural.
- The view volume is instead mapped to a canonical view volume (CVV) which is a cube that extends from -1 to $+1$ in each dimension, having center at the origin. The dimensions of the CVV facilitate a fast and efficient clipping.

Example of CVV:

CVV is related to Projection transformation i.e. Orthographic (parallel) projection and Perspective projection as **shown in the below figure:**

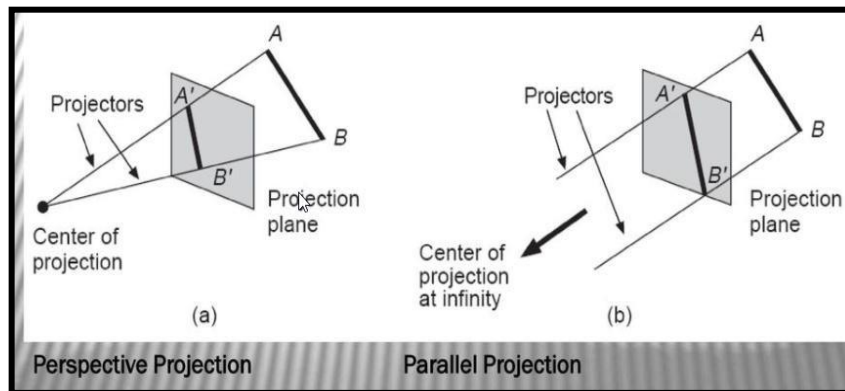


Fig 67 Perspective and Parallel (Orthographic) projection for CVV demonstrations

Note that **n** and **f** are typically given as distance which is always positive and because we are looking towards the $-z$ direction, the actual coordinates become **$-n$ and $-f$** .

Also note the changes in the z -direction. This makes objects further away from the camera to have larger z values. In other words, CVV is a left-handed coordinate system.

We need to map the box with corners at $(l, b, -n)$ and $(r, t, -f)$ to the $(-1, -1, -1)$ and $(1, 1, 1)$ of CVV.

SPECIFYING AN ARBITRARY 3D VIEW

- We can view the object from the side, or the top, or even from behind.
- Therefore, it is necessary to choose a particular view for a picture by first defining a view plane.
- A view plane is nothing but the film plane in a camera which is positioned and oriented for a particular shot of the scene.

- World coordinate positions in the scene are transformed to viewing coordinates, and then viewing coordinates are projected onto the view plane.
- The procedure to project from 3D to 2D given a finite view volume, will be as follows:
 - ❖ Apply a normalizing transform to get to the canonical view volume.
 - ❖ Clip against the canonical view volume
 - ❖ Project onto the view plane
 - ❖ Transform into viewport
- To specify an arbitrary view, we should be able to place the view plane anywhere in 3D. Specify the direction of the plane and its position in the world reference coordinate system (WRC).
 - ❖ A View Reference Point (VRP) which is to point on the plane.
 - ❖ A View Plane Normal (VPN) which is the normal vector to the plane.
 - ❖ A View Up Vector (VUP) which is a vector from which we determine which way is up.
- **Window on View plane:**
 - ❖ **PRP:** Projection Reference Point
 - ❖ **CW:** Center of Window
 - ❖ **COP:** Center of Projection (Perspective Projection)
 - ❖ **DOP:** Direction of Projection (Parallel Projection)
 - ❖ PRP and CW are used to determine COP and DOP
 - ❖ **Perspective:** $COP = PRP$
 - ❖ **Parallel:** $DOP = PRP - CW$
- **Coordinate Systems:**
 - ❖ **WC:** World Coordinates - normal, 3-space (x, y, z)
 - ❖ **VRC:** Viewing Reference Coordinates - defined by VRP, VPN and VUP

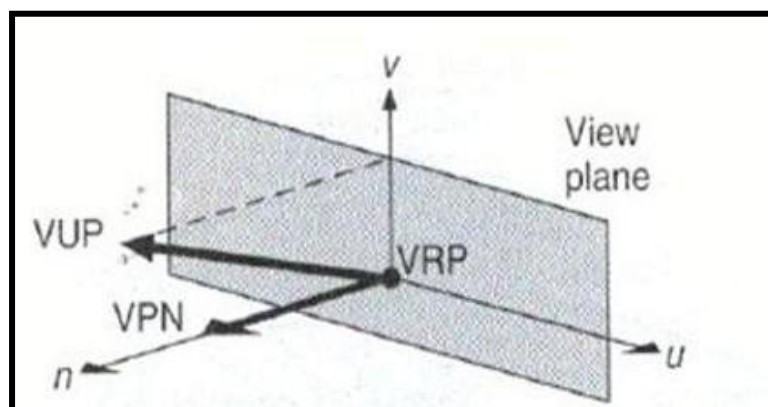


Fig 68: The view plane is defined by VPN and VRP, the v axis is defined by the projection of VUP along VPN onto the view plane. The u axis forms the right-handed VRC system with VPN and v.

- The notation (WC) or (VRC) is added to the table as a prompt of the coordinate system in which the viewing parameters is given. The form of the table is demonstrated for the default viewing specification used by PHIGS.
- The corresponding values for default view volume is given by,

Viewing parameter	Value	Comments
VRP(WC)	(0,0,0)	Origin
VPN(WC)	(0,0,1)	Z axis
VUP(WC)	(0,1,0)	Y axis
PRP(VRC)	(0.5,0.5,1.0)	
Window(VRC)	(0,1,0,1)	
Projection type	parallel	

Perspective Projections

- To obtain the front one-point perspective view of the house shown in **Figure** the position the center of projection (which can be thought of as the position of the viewer) at $x = 8$, $y = 6$, and $z = 84$. The x value is selected to be at the horizontal center of the house and the y value to correspond to the estimated eye level of a viewer standing on the (x, z) plane; the z value is arbitrary. In this case, z is removed 30 units from the front of the house ($z = 54$ plane). The window has been made quite large, to guarantee that the house fits within the view volume. All other viewing parameters have their default values, so the overall set of viewing parameters is as follows:

VRP(WC)	(0, 0, 0)
VPN(WC)	(0, 0, 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(8, 6, 84)
window(VRC)	(-50, 50, -50, 50)
projection type	perspective

- Although the image in **Figure** is indeed a perspective projection of the house, it is very small and is not centered on the view surface. We would prefer a more centered projection of the house that more nearly spans the entire view surface, as in **Figure** we can produce this effect more easily if

the view plane and the front plane of the house coincide.

- Now, because the front of the house extends from 0 to 16 in both x and y, a window extending from 1 to 17 in x and y produces reasonable results.

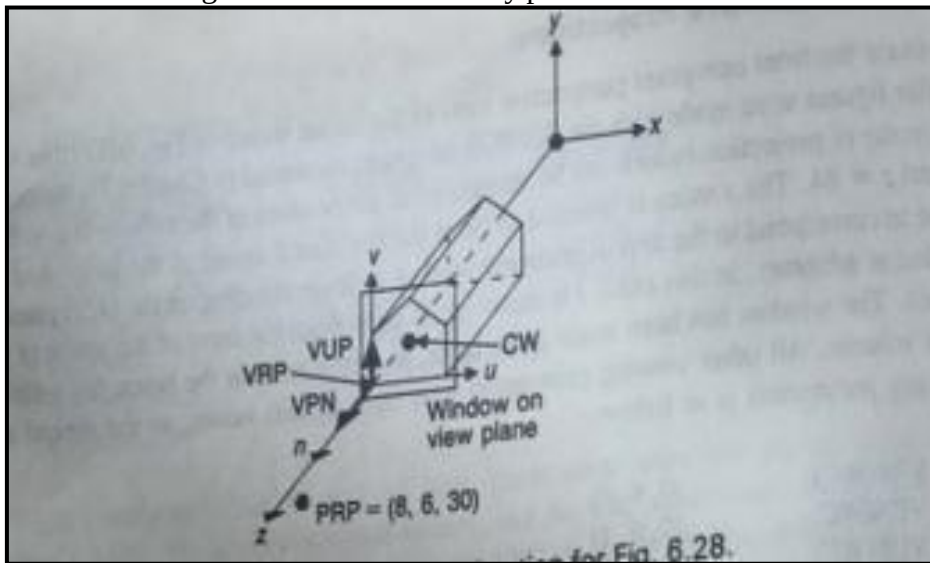


Fig 69 The viewing situation

Parallel Projections:

- We create a front parallel projection of the house (**Figure**) by making the direction of projection parallel to the z axis and direction of projection is determined by the PRP and by the center of the window.
- With the default VRC system and a window of (-1, 17,-1, 17), the center of the window is (8, 8, 0).
- The window is symmetrical about VRP, which implies that VRP, the origin of the VRC system, is the window's center. Placing PRP as specified means that the direction of projection, which is the vector from PRP to the window center, is $(0, 0, 0) - \text{PRP} = -\text{PRP}$, or its negative PRP.

Finite View Volumes

- In all the example, the view volume has been expected to be infinite.
- The front and back clipping planes, help to determine a finite view volume.
- These planes, both of which are parallel to the view plane, are at distances F and B respectively from the view reference point, measured from VRP along VPN.
- To avoid a negative view volume, we must ensure that F is algebraically greater than B.

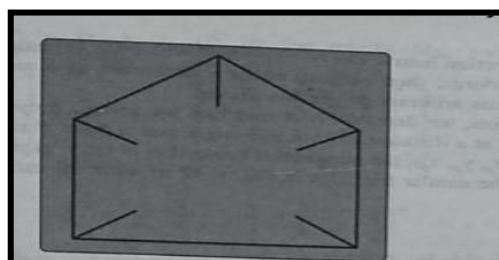


Fig 70 Perspective projection of the house with back clipping plane at $z = 31$.

- If a distance is given, then clipping against the corresponding plane is assumed; otherwise, it is not.
- If the front and back clipping planes are moved dynamically, the 3D structure object being viewed can often be discerned more readily than it can with a static view.

THE MATHEMATICS OF PLANAR GEOMETRIC PROJECTIONS

- Planar projections are the subset of 3D graphical projections constructed by linearly mapping points in three-dimensional space onto a projection plane. The projected point on the plane is chosen such that it is collinear with the corresponding three-dimensional point and the centre of projection. The lines connecting these points are commonly referred to as projectors.
- Type of planar geometric projection is obtained by the proper choice of a projection plane and a center of projection. The mathematical properties of each method of projection are related to the properties of related objects.
- Each type of projection provides variety of visual effects.
- Mathematically, planar projections are linear transformations acting on a point in three-dimensional space $a(x, y, z)$ to give a point $b(u, v)$ on the projection plane.
- These transformations consist of various compositions of the five transformations: **orthographic, projection, rotation, shear, translation and perspective.**
- For simplicity, we start by assuming that, in the perspective projection, the projection plane is normal to the z -axis at $z=d$, and that, in the parallel projection, the projection plane is the $z=0$ plane. Each of the projection can be defined by a 4×4 matrix. Because the projection matrix can be composed with transformation matrices, allowing two operations (transform, then project) to be represented as a single matrix.
- In arbitrary projection planes we derive 4×4 matrices for several projections, beginning with a projection plane at a distance d from the origin and a point $\mathbf{P}$ to be projected onto it.
 - This formulation allows d , the distance to the center of projection to tend to infinity.
 - M_{per} applies only in the special case in which the center of projection is at the origin; M_{ort} applies only when the direction of projection is parallel to the z axis.

COMBINED TRANSFORMATION MATRICES FOR PROJECTIONS AND VIEWING

After having all required matrices of projection and viewing, we can now combine all our transformation into one overall matrix, which will convert a point from world coordinates to Normalized Device Coordinates (NDC) while retaining a representation of the relative depth of the points.

$$M_{TOT} = A_{wv} * M_p * N$$

Where,

M_{TOT} : - Represents Combined transformation matrices for projection and viewing

A_{wv} : - Represents Metrics of real-world coordinates

M_p : - Represents Metrics of perspective transformation

N : - Represents Normalization Matrix

This will give us the view volume in its final configuration, called as **Canonical View Volume**.

COORDINATE SYSTEMS AND MATRICES

Co-ordinate System:

1. Screen coordinates
2. Local co-ordinate system or 3D modelling coordinate
3. Word coordinate
4. View reference coordinates
5. Normalized projection coordinates
6. 2D device coordinate

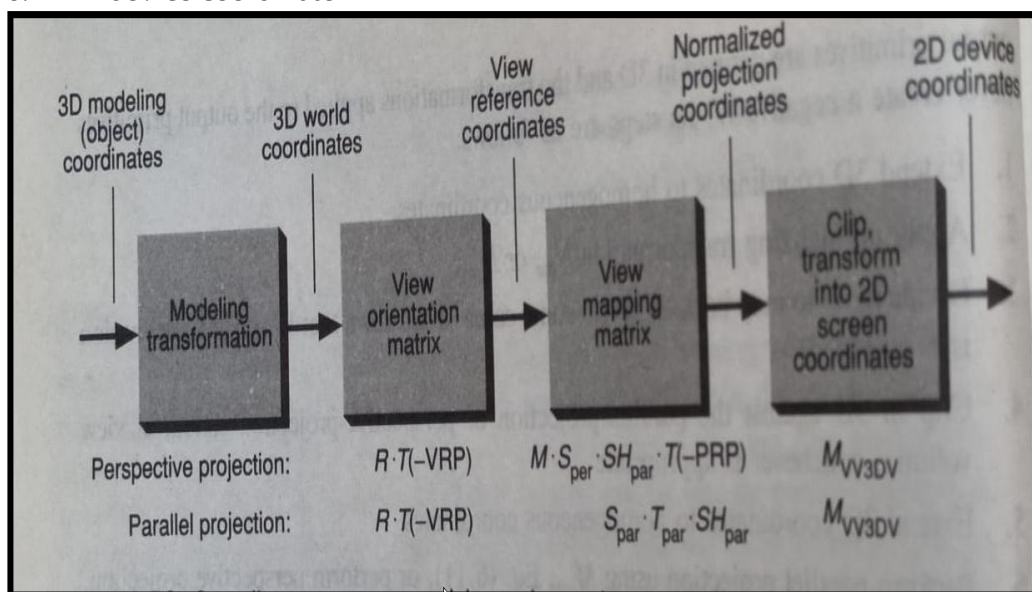


Fig 71 Coordinate systems and how they relate to one another.

The matrices underneath each stage effect the transformation applied at that stage for the perspective and parallel projections.

- Starting with the coordinate system that is utmost detached from the real display device, on the left of **Figure**, individual objects are well-defined in an object-coordinate system, PHIGS (**P**rogrammer's **H**ierarchical **I**nteractive **G**raphics **S**ystem) calls this as modeling-coordinate system; the term local coordinate system is also commonly used.
- Objects are transformed into the world-coordinate system, the system in which a scene or complete object is represented in the computer, by the modeling transformation. This system is sometimes called the problem-coordinate system or application-coordinate system.
- The view-reference coordinate system is used by PHIGS as a coordinate system to define a view volume. It is also called the (u, v, n) system, or the (u, v, VPN) system.
- The Core system used alike, but unidentified, left-handed system. The left-handed system is used so that, with the eye or camera at the origin looking toward +z, increasing values of z are beyond away from the eye, x is to the right, and y is up.
- Other packages, such as Pixar's RenderMan [PIXA88], place constraints on the view-reference coordinate system, requiring that the origin be at the center of projection and that the view plane normal be the z axis known as eye-coordinate system, Renderman and some other systems use the term camera-coordinate system.
- The first three steps of the perspective-projection normalizing transformation convert from the world-coordinate system into the eye-coordinate system. The eye coordinate system is sometimes left-handed.
- From eye coordinates, we next go to the normalized-projection coordinate system, or 3D screen coordinates, the coordinate system of the parallel-projection canonical view volume (and of the perspective-projection canonical view volume after the perspective transformation).
- The Basic system calls this system 3D normalized device coordinates. Sometimes, the system is called 3D logical device coordinates. The term normalized generally means that all the coordinate values are in either the interval [0, 1] or [-1, 1], whereas the term logical generally means that coordinate values are in some other prespecified range, such as [0, 1023], which is naturally defined to resemble some extensively available device's coordinate system.

- Projecting from 3D into 2D known as 2D device coordinate system, also called the normalized device-coordinate system, the image- coordinate system or the screen-coordinate system stated by RenderMan.
- Other terms used include screen coordinates, device coordinates, 2D device coordinates, and physical device coordinates (in contrast to the logical device coordinates stated earlier).
- RenderMan calls the physical fore of the space as raster coordinates.

CAMERA MODEL AND VIEWING PYRAMID

- In 3D computer graphics, the view frustum (also called viewing frustum) is the region of space in the modeled world that may appear on the screen; it is the field of view of the “notional camera”.
- The view frustum is typically obtained by taking a frustum- that is a truncation with parallel planes- of the pyramid of vision, which is the adaption of (idealized) cone of vision that a camera or eye would have to the rectangular viewports typically used in computer graphics.
- The exact shape of this region varies depending on what kind camera lens is being simulated, but typically it is a frustum of a rectangular pyramid (hence the name). The planes that cut the frustum perpendicular to the viewing direction are called the near plane and the far plane. Objects closer to the camera than the near plane or beyond the far plane are not drawn. Sometimes, the far plane is placed infinitely far away from the camera so all objects within the frustum are drawn regardless of their distance from the camera.

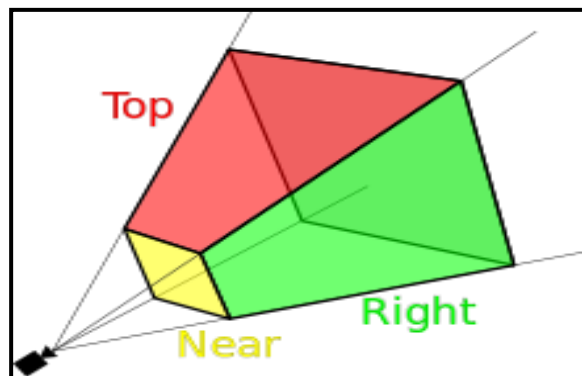


Fig 72 View frustum

View frustum culling is the process of removing objects that lie completely outside the viewing frustum from the rendering process. Rendering these objects would be a waste of time since they are not directly visible. To make culling fast, it is usually done using bounding volumes surrounding the objects rather than the objects themselves.

Module IV: Hidden surface elimination algorithms
4.1 Z-buffer.
4.2 Scan-line.
4.3 Sub-division.
4.4 Painter's algorithm.
4.5 Illumination Models: Diffuse reflection, Specular reflection, refracted light, texture surface patterns, Halftoning, Dithering.
4.6 Surface Rendering Methods: Constant Intensity method, Gouraud Shading, Phong Shading.

VISIBLE-SURFACE DETERMINATION

Techniques for efficient Visible-surface Algorithms

The techniques to perform visible surface algorithms efficiently are discussed in the following sections.

1. Coherence

The coherence is defined as the degree to which parts of an environment, or its projection exhibit local similarities. Such as similarities in depth, colour, texture and so on. To make algorithms more efficient we can exploit these similarities when we reuse calculations made for one part of the environment or a picture for other nearby parts, either without changes or with some incremental changes. Different kinds of coherence we can use in visible surface algorithms are.

- i. **Object coherence:** if one object entirely separates from another, comparisons may need to be done only between the two objects, and not between their components faces or edges.
- ii. **Face coherence:** Usually surface properties vary smoothly across a face. This allows the computations for one part of face to be used with incremental changes to the other parts of the face.
- iii. **Edge coherence:** The visibility of edge may change only when it crosses a visible edge or penetrates a visible face.
- iv. **Implied edge coherence:** If one planar face penetrates another their line of intersection can be determined from two points of intersection.
- v. **Area coherence:** A group of adjacent pixels is often belonging to the same visible face.
- vi. **Span coherence:** It refers to a visibility of face over a span of adjacent pixels on a scan line. It is special case area coherence.
- vii. **Scan line coherence:** The set of visible object spans determined for one scan line of an image typically changes very little from the set on the previous line.
- viii. **Depth coherence:** Adjacent parts of the same surface are typically same or very close depth. Therefore, once the depth at one point of the surface is determined, the depth of the points on the rest of the surface can be determined by at the simplest incremental calculation.
- ix. **Frame coherence:** Pictures of the same scene at two successive points in time are likely to be quite similar, except small changes in

object and view ports. Therefore, the calculations made for one picture can be reused for the next picture in a sequence.

2. Perspective Transformation:

Visible-surface determination is done in a 3D space prior to the projection into 2D that destroys the depth information needed for depth comparisons, and depth comparisons are typically done after the normalizing transformation.

Due to this projector are parallel to the Z axis in parallel projections or emanate from the origin in perspective projections.

In parallel projection, when $x_1 = x_2$ and $y_1 = y_2$ we can say that points are on the same projector.

However, in perspective projection we must perform four divisions: $x_1/z_1 = x_2/z_2$ and $y_1/z_1 = y_2/z_2$ to determine whether the points are on the same projector.

These divisions can be avoided by first transforming a 3D object into the 3D screen-coordinate systems, so that the parallel projection of the transformed object is the same as the perspective projection of the untransformed object.

3. Extents and bounding Volumes:

- a. Two objects with their projections and the rectangular screen extents surrounding the projections.
- b. It is easier to check the overlapping of extents than the projections, and we can say that when the extents are not overlapping then projections are also not overlapping. Therefore, extents must be compared first and then the projections must be compared only if the extents are overlapped. This avoids unnecessary comparisons in checking the overlapping of projections.
- c. Extents used to surround the object themselves rather than their projections; in this case, the extent becomes solid and are commonly known as bounding volumes. In this case extent can be used to check whether two objects are overlapped or not.
- d. Extent and bounding volumes are used to determine whether projector intersects an object or not.

4. Back-Face Culling

- a. When an object is approximated by a solid polyhedron, its polygonal faces completely enclose its volume. In such case, if none of the polyhedron's interior is exposed by the front clipping plane, then those polygons whose surface normal point away from the observer lie on a part of the polyhedron whose visibility is completely blocked by other closer polygons.
- b. Polygons (A, B, D, F) in gray are eliminated, whereas front-facing polygon (C, E, G, H) are retained.
- c. These invisible back-facing polygons can be eliminated from further processing. Such a technique to eliminate the back-facing polygon from further processing is known as Back-face Culling.

5. Spatial Partitioning:

- a. In this technique, subdivision rule is applied to break down a large problem into a number of smaller ones. This object and their projections are assigned to spatially coherent groups as a pre-processing step. This partitioning speed up the process of determining which object intersect with a projector. Because now it is necessary to test only the objects lying within the partitions which intersect with projector.
- b. When objects are unequally distributed in space the adaptive partitioning is more suitable. Because it allows variable size of each partition.

6. Hierarchy

- a. In hierarchical structure different levels are assigned to the object and there is a parent-child relationship between the objects.
- b. In this structure, each child is considered as a part of its parent. This allows to restrict the number of object comparison needed by a visible-surface algorithm.
- c. If the parent level object is fail to intersect, the lower level objects belongs to the parent do not need to be tested for intersection.

Object Space Method and Image Space Method:

Object Space method	Image Space method
1. It deals with object definition directly	1. It is a pixel-based method. It is concerned with the final image, what is visible within each raster pixel.
2. Here surface visibility is determined.	2. Here line visibility or point visibility is determined.
3. It is performed at the precision with which each object is defined, No resolution is considered.	3. It is performed using the resolution of the display device.
4. Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4. Calculations are resolution base, so the change is difficult to adjust.
5. These were developed for vector graphics system.	5. These are developed for raster devices.
6. Object-based algorithms operate on continuous object data.	6. These operate on object data.
7. Vector display used for object method has large address space.	7. Raster systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. There are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.

10. If the number of objects in the scene increases computation time also increases.	10. In this method complexity increase with the complexity of visible parts.

CATEGORIES OF ALGORITHMS

Back- face removal:

1. A polygon has two surfaces, a front and a back just as a piece of paper does. We might picture our polygons with one side painted light and the other painted dark. But the question is “how to find which surface is light or dark”.
2. When we are looking at the light surface, the polygon will appear to be drawn with counter clockwise pen motions, and when we are looking at the dark surface the polygon will appear to be drawn with clockwise pen motions.
3. Let us assume that all solid objects are to be constructed out of polygons in such a way that only the light surfaces are open to the air; the dark faces meet the material inside the object. This means that when we look at an object face from the outside, it will appear to be drawn counter clockwise.
4. If a polygon is visible, the light surface should face towards us and the dark surface should face away from us. Therefore, if the direction of the light face is pointing towards the viewer, the face is visible, otherwise the face is hidden and should be removed.
5. The direction of the light face can be identified by examining the result

$$N.V > 0$$

Where N: Normal vector to the polygon surface with cartesian components(A,B,C).

1V: A vector in the viewing direction from the eye position.
6. We know that, the dot product of two vector gives the product of the length of the two vectors times the cosine of the angle between them.
7. This cosine factor is important to us because if the vectors are in the same direction ($0 \leq \Theta < \pi/2$), then the cosine is positive, and the overall dot product is positive. However, if the direction is opposite ($\pi/2 < \Theta \leq \pi$), then the cosine and the overall dot product is negative.
8. If the dot product is positive, we can say that the polygon faces towards the viewer; otherwise it faces away and should be removed.
9. In case, if object description has been converted to projection coordinates and our viewing direction is parallel to the viewing z axis, then $V=(0, 0, V_z)$ and $V.N = V_z C$
10. So that we only have to consider the sign of C, the Z component of the normal vector N. Now, if the z component is positive, then the polygon faces towards the viewer, if negative, it faces away.

Scan Line Algorithm:

A scan line method of hidden surface removal is another approach of image space method. It is an extension of the scan line algorithm for filling polygon interiors. Here, the algorithm deals with more than one surfaces. As each scan line is processed, it examines all polygon surfaces intersecting that line to determine which are visible. It does the depth calculation and finds which polygon is nearest to the view plane. Finally, it enters the intensity value of the nearest value of the nearest polygon at that position into the frame buffer.

1. We know that scan line algorithm maintains the active edge list. This active edge list contains only edges that cross the current scan line, sorted in order of increasing x . The scan to indicate whether a position along a scan line is inside or outside of the surface. Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned ON, and at the rightmost boundary, it is turned OFF.

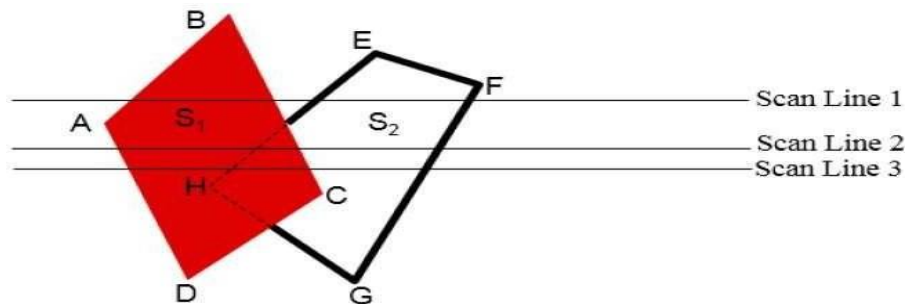


Fig 73: Example 1 Scan Line Algorithm

2. **Example 1:** The figure illustrates the scan line method for hidden surface removal. The active edge list for scan line 1 contains the information for edges AD, BC, EH and FG. For the positions along this scan line between edges AD and BC, only the flag for surface S_1 is ON. Therefore, no depth calculations are necessary, and intensity information for surface S_1 is entered the frame buffer. Similarly, between edges EH and FG, only the flag for surface S_2 is ON and during that portion of scan line intensity information for surface S_2 is entered the frame buffer.
3. For scan line, the active edge list contains edges AD, EH, BC, FG. Along the scan line 2 from edge AD to edge EH, only the flag for surface S_1 is ON. However, between edges EH and BC, the flags for both surface are ON. In this portion of scan, line 2, the depth calculations are necessary. Here we have assumed that the depth of S_1 is less than the depth of S_2 and hence the intensities of surface S_1 are loaded into the frame buffer. Then for edge BC to edge FG portion of scan line 2 intensities of surface S_2 are entered the frame buffer because buffer during that portion only flag for S_2 is ON.
4. This algorithm is based on the Image-space method and concept of coherence. As its name suggests itself Scan-line algorithm, so it processes one line at a time rather than processing one pixel(a point on raster display) at a time.

5. **Edge list table:** This list maintains the record of all the edges by storing their endpoint coordinates. The x-coordinate that we choose, whose Y-coordinate = Ymin.
6. **Active edges table(list):** This table contains all those edges of the polygon that are intersected(crossed) by the current scan-line. The edges are dropped into the table in a sorted manner(Increasing value of x).
7. **Polygon table(list):** This list consists of:
 - a. Polygon Id.
 - b. Plane equation.
 - c. Color Information of the surface.
 - d. Flag of surface(on/off)].
1. **First, examine the scanline(S_1), whose,** Active edge table (Aet) contains [AD,BC,RS,PQ], and the flag is set to “on” for surface(S_1). In addition it contains surface(S_2) and the flag set “off” for the region between BX and RX as it’s an outer region of the polygon’s surface and not to be projected at view-port(display devices). Now drop the color-intensities of the corresponding surfaces into the frame buffer(refresh buffer).
2. **Then, process the scanline(S_2), whose,** Active edge table (Aet) contains[AD,BC,RS,PQ], and the flag is set to “on” for surface (ABCD) and surface(PQRS). Both of the polygons surfaces are overlapping each other. So for this overlapped region that which of the surface intensity should be taken into account? calculate the depth (Z_{min}) of both surface(S_1) and surface(S_2) of the overlapped portion, of surface intensity . Next, If $depth(S_1) > depth(S_2)$, then the Flag of surface S_1 =” on” and intensity of surface S_1 will be considered else S_2 , now Drop the color-intensities of the corresponding surfaces whose flag is set to on into the frame buffer(refresh buffer).
3. As Scanline(S_3) is passing through the same portion from where Scanline(S_2) is passing, S_3 also has the same Active edge table(Aet) components as S_2 has and no need to calculate the $depth(S_1)$ and $depth(S_2)$ again so S_3 can take the advantage of the concept of Coherence.

Z-Buffer Algorithm:

1. One of the simplest and commonly use image space approach to eliminate hidden surface is the **Z-buffer** or **depth buffer** algorithm. It is developed by Catmull. This algorithm compares surface depths at each pixel position on the projection plane.
2. The surface depth is measured from the view plane along the z axis of a viewing system. When object description is converted to projection coordinates (x, y, z), each pixel position on the view plane is specified by x and y coordinate, and z values gives the depth information. Thus, object depths can be compared by comparing the z-values.

3. The Z-buffer algorithm is usually implemented in the normalized coordinates, so that z values range from 0 at the back-clipping plane to 1 at the front clipping plane. The implementation requires another buffer called Z-buffer along with the frame buffer memory required for raster display devices. A Z-buffer is used to store depth values for each (x, y) position as surfaces are processed, and the frame buffer stores the intensity values for each position.
4. At the beginning the z -value at the back-clipping plane, and the frame buffer is initialized to the background colour. Each surface listed in the display file is then processed, one scan line at a time calculating the depth (z -value) at each (x, y) pixel position.
5. The calculated depth value is compared to the value previously stored in the Z-buffer at that position. If the calculated depth value is greater than the value stored in the Z-buffer, the new depth value is stored, and the surface intensity at that position is determined and placed in the same xy location in the frame buffer.
6. For example, among three surfaces, surface S_1 has the smallest depth at view position (x, y) and the highest z value. So, it is visible at that position.

Algorithm:

1. Initialize the Z-buffer and frame buffer so that for all buffer positions $z\text{-buffer}(x,y)=0$ and $\text{Frame-buffer}(x,y)=I_{\text{background}}$
2. During scan conversion process, for each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
3. Calculate z -value for each (x,y) position on the polygon
 If $z > z\text{-buffer}(x,y)$ then
 $z\text{-buffer}(x,y)=z$,
 $\text{Frame-buffer}(x,y)=I_{\text{surface}}(x,y)$.
4. Stop.
 (After processing of all surfaces, the z -buffer contains depth values for the visible surfaces and the frame buffer contain the corresponding intensity values for those surfaces.)

Advantages:

1. It is easy to implement
2. It can be implemented in hardware to overcome the speed problem.
3. Since the algorithm processes objects one at a time, the total number of polygons in a picture can be arbitrarily large.

Disadvantages:

1. It requires an additional buffer and hence the large memory.
2. It is time consuming process as it requires comparison for each pixel instead of for the entire polygon.

Warnock's Algorithm (Area Subdivision Algorithm):

An interesting approach to the hidden-surface problem was developed by Warnock. He Developed area subdivision algorithm which subdivides each area into four equal squares.

At each stage in the recursive-subdivision process, the relationship between projection of each polygon and the area of interest is checked for four possible relationships:

1. Surrounding polygon – One that completely encloses the (shaded) area of interest.
2. Overlapping or Intersecting Polygon – One that is partly inside and partly outside the area.
3. Inside or Contained Polygon – One that is completely inside the area.
4. Outside or Disjoint Polygon- One that is completely outside the area.

After checking four relationship we can handle each relationship as follows:

1. If all the polygons are disjoint from the area, then the background colour is displayed in the area.
2. If there is only one intersecting or only one contained polygon, then the area is first filled with background colour, and then the part of the polygon contained in the area is filled with colour of polygon.
3. If there is a single surrounding polygon, but no intersecting or contained polygons, then the area is filled with the colour of the surrounding polygon.
4. If there are more than one polygon intersecting, contained in, or surrounding the area then we must do some more processing.

Algorithm:

1. Initialize the area to be the whole screen.
2. Create the list of polygons by sorting them with their z-values of vertices. Don't include disjoint polygons in the list because they are not visible.
3. Find the relationship of each polygon.
4. Perform the visibility decision test
 - a. If all the polygons are disjoint from the area, then fill area with background colour.
 - b. If there is only one intersecting or only one contained polygon

then first fill entire area with background colour and then fill the part of the polygon contained in the area with the colour of polygon.

- c. If there is a single surrounding polygon, but no intersecting or contained polygons, then fill the area with the colour of the surrounding polygon.
 - d. If the surrounding polygon is closer to the viewpoint than all other polygons, so that all other polygons are hidden by it, fill the area with the colour of the surrounding polygon.
 - e. If the area is the pixel (x, y) and neither a, b, c, nor d apply, compute the z coordinate at pixel (x, y) of all polygons in the list. The pixel is then set to colour of the polygon which is closer to the viewpoint.
5. If none of the above tests are true, then subdivide the area and go to step 2.

Advantages:

1. It follows the divide-and-conquer strategy; therefore, parallel computers can be used to speed up the process.
2. Extra memory buffer is not required.

Binary Space Partition (BSP) Trees:

Binary space partitioning is used to calculate visibility. To build the BSP trees, one should start with polygons and label all the edges. Dealing with only one edge at a time, extend each edge so that it splits the plane into two. Place the first edge in the tree as root. Add subsequent edges based on whether they are inside or outside. Edges that span the extension of an edge that is already in the tree are split into two and both are added to the tree.

Painter's Algorithm (Depth Sorting Method):

- Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions –
 - First, the surfaces are sorted in order of decreasing depth.
 - Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.
- The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the painter's algorithm. The following figure shows the effect of depth sorting –The algorithm begins by sorting by depth. For example, the initial “depth” estimate of a polygon may be taken to be the closest z value of any vertex of the polygon.
- Let us take the polygon P at the end of the list. Consider all polygons Q whose z-extents overlap P's. Before drawing P, we

make the following tests. If any of the following tests is positive, then we can assume P can be drawn before Q.

Visible-Surface Ray Tracing:

- Ray tracing is an image-based algorithm. For every pixel in the image, a ray is cast from the center of projection through that pixel and into the scene. The colour of the pixel is set to the colour of the object that is first encountered.

COMPARISON OF THE METHODS

- Hardware available? Use depth-buffer, possibly in combination with back-face elimination or depth-sort for part of scene.
- If not, choose dependent on complexity scene and type of objects:
- Simple scene, few objects: depth-sort
- Quadratic surfaces: ray-casting
- Otherwise: depth-buffer
- Many additional methods to boost performance (kD-trees, scene decomposition, etc.)



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

**The Motto of the University
(SEWA)**

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



**Bachelor of Computer Applications (BCA)
Course Name: Object Oriented Programming Lab
Course Code: BCA-4-03P**

**ADDRESS: C/28, THE LOWER MALL, PATIALA-147001
WEBSITE: www.psou.ac.in**



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME COORDINATOR :

Dr. Monika Pathak

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

PROGRAMME CO-COORDINATOR :

Dr. Gaurav Dhiman

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

COURSE COORDINATOR :

Dr. Karan Sukhija (Assistant Professor)
School of Sciences and Emerging Technologies
JGND PSOU, Patiala



**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA**
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

Bachelor of Computer Applications (BCA)
Semester IV
BCA-4-03P: Object Oriented Programming Lab

Total Marks: 50
External Marks: 35
Internal Marks: 15
Credits: 2
Pass Percentage:
40%

Course: Object Oriented Programming Lab	
Course Code: BCA-4-03P	
Course Outcomes (COs)	
After the completion of this course, the students will be able to:	
CO1	Develop the ability to apply OOP fundamentals in creating well-structured and readable code.
CO2	Develop Programs for file handling.
CO3	Develop Programs for Operator Overloading.
CO4	Gain practical experience in implementing OOP concepts such as classes, objects, inheritance, and polymorphism in programming assignments.
CO4	Gain proficiency in using programming languages that support OOP to develop applications and solve real-world problems.

Detailed List of Programs:

Programme No.	Name of Program
P1	Create a class with attributes and methods, and then create objects of that class to demonstrate basic OOP principles.
P2	Create a base class and one or more derived classes to demonstrate inheritance and the use of base class members in derived classes.
P3	Create a base class with a virtual function, override the function in a derived class, and demonstrate runtime polymorphism by calling the function through base class pointers.
P4	Create a class with private data members and public member functions to demonstrate encapsulation and data hiding.
P5	Create a class with a constructor and destructor to demonstrate object

	initialization and cleanup.
P6	Overload arithmetic or comparison operators for a class to demonstrate operator overloading.
P7	Create a class with multiple functions of the same name but different parameters to demonstrate function overloading.
P8	Create an abstract class with one or more pure virtual functions to demonstrate abstract classes and interfaces.
P9	Create a program that allows the user to add, delete, modify, and display student records. Use file handling to store and retrieve student data.

Students/Learners can implement Object Oriented Programming (OOP) concepts using any language like C++ or Java or Python

(Part A)

- 1) Program to swap two values using pointers and reference variables.
- 2) Program to find Largest and smallest of four numbers using inline functions.
- 3) Program to check whether the given number is prime or not using function overloading and also use default arguments.
- 4) Program to find factorial of a number using function overloading (use both direct and recursive methods to find factorial).
- 5) Program to accept cricket players name, total runs, and total matches and print these details with batting average (Use array of objects).
- 6) Program in which Create a class to hold information of a husband and another for the wife. Using friend function find the total salary of the family.
- 7) Program to demonstrate static members.
- 8) Program to create a database for a bank account contains Name, Account no, Account type, Balance, Including the following a) Constructors b) destructors c) default constructors d) Input and output functions.

(Part B)

- 9) Program to increment the given date using ++ operator (unary operator).
- 10) Program to overload the following operators:
 - a) Binary operator '+' to concatenate 2 strings
 - b) compare to strings using == operator
- 11) Program in which Create a base class for a stack and implement push and pop operation. Include a derived class to check for stack criteria such as a) stack empty b) stack full c) stack overflow d) stack underflow.
- 12) Program to illustrate hybrid inheritance.
- 13) Program to sort n names using pointer sort.
- 14) Program to demonstrate Virtual function.
- 15) Program to create a database using concepts of files for a student including the fields:

Student-name, Student's Register No, Student's Attendance (overall % of attendance) and enter data for n students and output the same in proper format.

16) Program to accessing a particular record in an employee file.

Practice Exercise:

- Write a C++ Program to display Names, Roll No., and grades of 3 students who have appeared in the examination. Declare the class of name, Roll No. and grade. Create an array of class objects. Read and display the contents of the array.
- Write a C++ program to declare Struct. Initialize and display contents of member variables.
- Write a C++ program to declare a class. Declare pointer to class. Initialize and display the contents of the class member.
- Given that an EMPLOYEE class contains following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary and print data members.
- Write a C++ program to read the data of N employee and compute Net salary of each employee (DA=52% of Basic and Income Tax (IT) =30% of the gross salary).
- Write a C++ to illustrate the concepts of console I/O operations.
- Write a C++ program to use scope resolution operator. Display the various values of the same variables declared at different scope levels.
- Write a C++ program to allocate memory using new operator.
- Write a C++ program to create multilevel inheritance. (Hint: Classes A1, A2, A3)
- Write a C++ program to create an array of pointers. Invoke functions using array objects.
- Write a C++ program to use pointer for both base and derived classes and call the member function. Use Virtual keyword.

Solutions:

```

/* 1. Program to swap two values using pointers and
reference variables */

#include<iostream.h>
#include<conio.h>
void swappoint(int *,int *);
void swapref(int &,int &);

void main()
{
int a,b;
clrscr();
cout<<"Enter two numbers"<<endl;
cin>>a>>b;
cout<<"\nValues of A and B before calling function"<<endl;
cout<<"A="<<a<<" B="<<b;
swappoint(&a,&b);
cout<<"\nValues of A and B after calling swappoint()"<<endl;
cout<<"A="<<a<<" B="<<b;
swapref(a,b);
cout<<"\nValues of A and B after calling swapref()"<<endl;
cout<<"A="<<a<<" B="<<b;
getch();
}

void swappoint(int *l,int *m)
{
int k;
k=*l;
*l=*m;
*m=k;
}

void swapref(int &l,int &m)
{
int k;

```

```

k=l;
l=m;
m=k;
}

/* 2. Program to find Largest and smallest of four numbers
   Using inline functions. */
#include<iostream.h>
#include<conio.h>
inline int largest(int x, int y)
{
return(x>y?x:y);
}
inline int smallest(int x, int y)
{
return(x<y?x:y);
}

void main()
{
clrscr();
int a,b,c,d;
cout<<"Enter any four numbers:";
cin>>a>>b>>c>>d;
int lar=largest(largest(a,b),largest(c,d));
cout<<"\n Largest of the given four numbers is:"<<lar;
int small=smallest(smallest(a,b),smallest(c,d));
cout<<"\n Smallest of the given four numbers is:"<<small;
getch();
}

/* 3. Program to check whether the given number is prime or
   not using function overloading and also use default
   arguments.*/

```



```

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void prime(), prime(int,int=1);

void main()
{
    int choice, n;

    while(1)
    {
        clrscr();
        cout<<"\n1.Prime number checking without parameters";
        cout<<"\n2.Prime number ckecking with parameters";
        cout<<"\n3.Exit";
        cout<<endl<<"\nEnter your choice:";
        cin>>choice;
        switch(choice)
        {
            case 1:prime();
                    getch();
                    break;
            case 2:cout<<"Enter any number:";
                    cin>>n;
                    prime(n);
                    getch();
                    break;
            case 3:exit(0);
                    break;
            default: cout<<"\nInvalid Choice";
                    getch();
        }
    }
}

void prime()

```

```

{
    int n, flag=1;
    cout<<"Enter any number:";
    cin>>n;
    for(int i=2;i<n/2;i++)
        if(n%i==0)
        {
            flag=0;
            break;
        }
    if(flag==1)
        cout<<endl<<"The Given Number "<<n<<" is a prime number";
    else
        cout<<endl<<"The Given Number "<<n<<" is not a prime
            number";
}

void prime(int n,int flag)
{
    for(int i=2;i<n/2;i++)
        if(n%i==0)
        {
            flag=0;
            break;
        }
    if(flag==1)
        cout<<endl<<"The Given Number "<<n<<" is a prime number";
    else
        cout<<endl<<"The Given Number "<<n<<" is not a prime
            number";
}

```

```

/* 4.Program to find factorial of a number using
    function overloading (use both direct and recursive
    methods to find factorial) */

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void fact();
long int fact(int);

void main()
{
    int choice, n;
    for(;;)
    {
        clrscr();
        cout<<"1.Factorial of a number without parameters"<<endl;
        cout<<"2.Factorial of a number with parameters"<<endl;
        cout<<"3.Exit"<<endl;
        cout<<endl<<"Enter your choice:";
        cin>>choice;
        switch(choice)
        {
            case 1:fact();
                getch();
                break;
            case 2:cout<<"Enter the number:";
                cin>>n;
                cout<<"\n The factorial of "<<n<<" is "<<fact(n);
                getch();
                break;
            case 3:exit(0);
            default:cout<<"\n Invalid choice";
                getch();
        }
    }
}

```

```

}

void fact()
{
    int n,i;
    cout<<endl<<"Enter the number:";
    cin>>n;
    long int fact=1;
    for(i=1;i<=n;i++)
        fact=fact*i;
    cout<<endl<<"The Factorial of "<<n<<" is "<<fact;
}

long int fact( int n)
{
    if(n==0)
        return 1;
    else
        return n*fact(n-1);
}

/* 5.Program to accept cricket players name, total
    runs, and total matches and print these details with
    batting average(Use array of objects). */

#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<iomanip.h>

class cricket
{
    char name[30];
    float truns;

```

```

float tmat;
float avg;
public:
void get_data();
void put_data();
};

void cricket::get_data()
{
cout<<"Enter the player name:"<<endl;
cin>>name;
cout<<"Enter total runs of the player "<<name<<":"<<endl;
cin>>truns;
cout<<"Enter total matches played by "<<name<<":"<<endl;
cin>>tmat;
avg=truns/tmat;
}

void cricket::put_data()
{
cout.setf(ios::left,ios::adjustfield);
cout<<setw(10)<<name<<"\t"<<setw(10)<<truns<<"\t"
<<setw(10)<<tmat<<"\t"<<setw(10)<<setprecision(2)<<avg<<endl;
}

void main()
{
clrscr();
int n;
cricket e[' '];
cout<<"Enter number of players:";
cin>>n;
for(int i=1;i<=n;i++)
{
cout<<"Enter the details of player "<<i<<endl;
e[i].get_data();
}
}

```

```

clrscr();
cout<<"-----"
        ----- "<<endl;
cout<<"Player-Name\tTotal-Runs\tTotal-Matches\tBatting-
        Average\n";
cout<<"-----"
        -----"<<endl;
for(i=1;i<=n;i++)
{
e[i].put_data();
}
cout<<"-----"
        -----";
getch();
}

/* 6. Program in which Create a class to hold
        information of a husband and another for the wife. Using
        friend function find the total salary of the family. */

#include<iostream.h>
#include<conio.h>

class wife;

class husband
{
char name[10];
float sal;

public:

void read()
{
cout<<"\nEnter Name and Salary of Husband:" ;
cin>>name>>sal;
}

```

```

void print()
{
cout<<"\nName:"<<name<<"\nSalary:"<<sal<<endl;
}
friend float total(husband,wife);
};

class wife
{
char name[10];
float sal;

public:

void read()
{
cout<<"\nEnter Name and Salary of Wife:";
cin>>name>>sal;
}

void print()
{
cout<<"\nName:"<<name<<"\nSalary:"<<sal<<endl;
}
friend float total(husband,wife);
};

float total(husband h,wife w)
{
return(h.sal+w.sal);
}

void main()
{
husband h;
wife w;

```

```

clrscr();
h.read();
w.read();
clrscr();
cout<<"\nHusband Details:";
h.print();
cout<<"\nWife Details:";
w.print();
cout<<"\nTotal Monthly Income of the Family:"<<total(h,w);
getch();
}

```

```

/* 7. Program to demonstrate static members */

```

```

#include<iostream.h>
#include<conio.h>

```

```

class count
{
private :
static int number;
int idno;
public:

```

```

void assignidno()
{
idno=++number;
}

```



```
void display()
{
cout<<"object number : "<<idno<<endl;
}

static void displaycount()
{
cout<<"Number of objects created so far : "<<number<<endl;
}
};

int count::number;

void main()
{
count obj1;
clrscr();
cout<<"\n Program to demonstrate static members \n\n";
obj1.assignidno();
count::displaycount();
count obj2,obj3;
obj2.assignidno();
obj3.assignidno();
count::displaycount();
obj1.display();
obj2.display();
obj3.display();
getch();
}
```

/* 8. Program to create a database for a bank account contains Name, Account no, Account type, Balance, Including the following a) Constructors b) destructors c) default constructors d) Input and output functions. */

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<iomanip.h>
class bank
{
    private:
        char name[20];
        int ano;
        char atype[2];
        float balance;
    public:
        bank() {}
        bank(char na[], int no, char type[], float bal)
        {
            strcpy(name, na);
            ano=no;
            strcpy(atype, type);
            balance=bal;
        }
        void readdata()
        {
            cout<<endl<<"Name :";
            cin>>name;
            cout<<endl<<"Account Number :";
            cin>>ano;
            cout<<endl<<"Account type :";
            cin>>atype;
            cout<<endl<<"Balance :";
            cin>>balance;
        }
}
```

```

void writedata()
{

cout<<endl<<setw(15)<<name<<setw(7)<<ano<<setw(9)<<atype
<<setw(15)<<balance;
}
~bank()
{
}
};

void main()
{
    int n;
    bank cust[' '],cust1("Ashok",1220,"SB",10000.00);
    clrscr();
    cout<<"Enter number of customers:";
    cin>>n;
    for(int i=1;i<=n;i++)
        cust[i].readdata();
    clrscr();
    cout.setf(ios::left,ios::adjustfield);
    cout<<"\n \n Database for Bank Account \n \n ";
    cout<<"Name          Number Type      Balance \n";
    cust1.writedata();
    for(i=1;i<=n;i++)
        cust[i].writedata();
    getch();
}

```

PART-B

```
/*9 Program to increment the given date using ++ operator  
(unary operator)*/
```

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
class udate
```

```
{
```

```
int day,month,year;
```

```
public:
```

```
void read()
```

```
{
```

```
cin>>day>>month>>year;
```

```
}
```

```
void write()
```

```
{
```

```
cout<<day<<"/"<<month<<"/"<<year;
```

```
}
```

```
void operator ++();
```

```
};
```

```
void udate :: operator ++()
```

```
{
```

```
day++;
```

```
if(day>31)
```

```
{
```

```
    if(month==12)
```

```
    {
```

```
        day=1;month=1;year++;
```

```
    }
```

```
    else
```

```
    {
```

```

        day=1;
        month++;
    }
}
if ( (month==4 || month==6 || month==9 || month==11) && (day>30) )
{
    day=1; month++;
}

if (month==2)
{
    if (year%4==0 && day>29)
    {
        day=1;
        month++;
    }
    else
        if (year%4!=0 && day >28)
        {
            day=1;
            month++;
        }
}
}

void main()
{
    udate d1;
    clrscr();
    cout<<"Enter the date:";
    d1.read();
    cout<<"\n The given date is:";
    d1.write();
    ++d1;
    cout<<endl<<"\n The incremented date is : ";
    d1.write();
    getch();
}

```

```

}

/* 10. program to overload the following operators

    a) Binary operator '+' to concatenate 2 strings
    b) compare to strings using == operator */

#include<iostream.h>
#include<string.h>
#include<conio.h>

class string
{
private:
    char str[80];
public:

    void getstring()
    {
        cin.getline(str,80);
    }

    void display()
    {
        cout<<str;
    }

    string operator +(string);
    int operator ==(string);
};

string string::operator +(string ss)
{
    string temp;
    strcpy(temp.str,str);
    strcat(temp.str,ss.str);
    return temp;
}

```

```

int string::operator ==(string ss)
{
return(strcmp(str,ss.str));
}

void main()
{
string s1,s2,s3;
clrscr();
cout<<"Enter the first string :";
s1.getstring();
cout<<endl<<"Enter the second string :";
s2.getstring();
s3=s1+s2;
clrscr();
cout<<endl<<"\n The first string is:";
s1.display();
cout<<endl<<"\n The second string is :";
s2.display();
cout<<endl<<"\n The concatenation of two string :";
s3.display();
if(s1==s2)
cout<<"\n The given two strings  are not equal";
else
cout<<"\n The given two strings are equal";
getch();
}

```

/* 11.Program in which Create a base class for a stack and implement push and pop operation. Include a derived class to check for stack criteria such as a) stack empty b) stack full c) stack overflow d) stack underflow. */

```

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>

```

```
#include<stdio.h>
#define max 5

class stack
{
protected:
    int st[max];
    int top;
public:
    stack()
    {
        top = -1;
    }

    void push()
    {
        int item;
        cout<<"enter the item to push:";
        cin>>item;
        top++;
        st[top]=item;
    }

    int pop()
    {
        int item;
        item=st[top];
        top--;
        return(item);
    }

    void display()
    {
        cout<<"\n the stack contains \n"<<endl;
        for(int i=top;i>=0;i--)
            cout<<st[i]<<endl;
    }
}
```



```

};

class stack_cond:public stack
{
public:

void push()
{
if(top==max-1)
cout<<"\n stack over flow"<<endl;
else
{
stack::push();
if(top == max-1)
cout<<endl<<" after push stack is full"<<endl;
}
}

int pop()
{
int item;
if(top == -1)
{
cout<<endl<<"stack underflow"<<endl;
return(0);
}
else
{
item=stack::pop();
if(top == -1)
cout<<endl<<"after pop stack is empty"<<endl;
return(item);
}
}

void display()
{

```

```

if(top == -1)
cout<<"stack is empty"<<endl;
else
stack::display();
}
};

void main()
{
stack_cond s;
int choice;
cout<<"\n program for stack using inhertance \n\n";
do
{
clrscr();
cout<<"1. push \n";
cout<<"2. pop \n";
cout<<"3. display \n";
cout<<"4. exit \n";
cout<<" \n enter your choice:";
cin>>choice;
switch(choice)
{
case 1: s.push();
s.display();
break;
case 2: s.display();
choice=s.pop();
if(choice!=0)
cout<<endl<<"the items "<<choice<<" is popped from the top
of stack"<<endl;
break;
case 3: s.display();
break;
case 4: exit(0);
}
cout<<"press any key to continue....."<<endl;

```

```

getch();
}
while(choice!=4);
}

/* 12. Program to illustrate hybrid inheritance */

#include<iostream.h>
#include<conio.h>

class student
{
    int regno;
    char name[20];
public:
    void input()
    {
        cout<<"Enter Register Number & Name:\n";
        cin>>regno>>name;
    }
    void output()
    {
        cout<<"\nName:"<<name;
        cout<<"\nRegno:"<<regno;
    }
};

class test: public student
{
    protected:
        int m1,m2,m3;
    public:
        void input()
        {
            cout<<"\nEnter 3 subject marks\n";

```

```

        cin>>m1>>m2>>m3;
    }
    void output()
    {
        cout<<"\nSubject1 Marks:"<<m1;
        cout<<"\nSubject2 Marks:"<<m2;
        cout<<"\nSubject3 Marks:"<<m3;
    }
};

class sports
{
    protected:
        int wt;
    public:
        void input()
        {
            cout<<"Enter Sports Weightage\n";
            cin>>wt;
        }

        void output()
        {
            cout<<"\nSports Weightage:"<<wt;
        }
};

class result: public test,sports
{
    int total;
    public:

    void caltotal()
    {
        student::input();
        test::input();
        sports::input();
    }
};

```

```

        total=m1+m2+m3+wt;
        clrscr();
        student::output();
        test::output();
        sports::output();
        cout<<"\nTotal marks:"<<total;
    }
};

```

```

void main()
{
    clrscr();
    result r;
    r.caltotal();
    getch();
}

```

/* 13. Program to sort n names using pointer sort */

```

#include<iostream.h>
#include<conio.h>
#include<string.h>

```

```

class student
{
private:
    char name[15];
public:

```

```

    void getdata()
    {
        cin>>name;
    }

```

```

    void putdata()

```

```

{
cout<<name<<endl;
}

char* getname()
{
return name;
}
};

void main()
{
int n;
void sort(student**,int);
student *stu[10];
clrscr();
cout<<"Enter the total number of students:";
cin>>n;

for(int i=0;i<n;i++)
{
cout<<"\n Enter the name of the student:"<<i<<endl;
stu[i]=new student;
stu[i]->getdata();
}
clrscr();
cout<<"\n Before sorting : \n";
cout<<"\n The names are :\n";
for(i=0;i<n;i++)
stu[i]->putdata();
sort(stu, n);
cout<<"\n After sorting:\n";
cout<<"\n The names are:\n";
for(i=0;i<n;i++)
stu[i]->putdata();
getch();
}

```

```

void sort(student** s,int n)
{
void swap(student**, student**);
for(int i=0;i<n;i++)
for(int j=i+1;j<n;j++)
swap(s+i, s+j);
}

void swap(student** s1, student** s2)
{
if(strcmp((*s1)->getname(), (*s2)->getname())>0)
{
student *temp = *s1;
*s1 = *s2;
*s2 = temp;
}
}

/* 14. Program to demonstrate Virtual function.*/

#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>

class media
{
protected:
char title[20];
double price;
public:
media(char *s,double p)
{
strcpy(title,s);
price=p;
}
}

```

```

    }
    virtual void display()=0;
};

class book: public media
{
    int noofpages;
public:
    book(char *s,double a,int p):media(s,a)
    {
        noofpages=p;
    }
    void display()
    {
        cout<<"\nTitle:"<<title;
        cout<<"\nPages:"<<noofpages;
        cout<<"\nPrice:"<<price;
    }
};

class tape: public media
{
    int playtime;
public:
    tape(char *s,double a,int pt): media(s,a)
    {
        playtime=pt;
    }
    void display()
    {
        cout<<"\nTitle:"<<title;
        cout<<"\nPlaytime:"<<playtime<<" Minutes";
        cout<<"\nPrice:"<<price;
    }
};

void main()

```



```

{
    clrscr();
    char t[100];
    double pr;
    int pt,nop;
    media *m;
    cout<<"Enter Title,NoofPages & price of a Book:\n";
    gets(t);
    cin>>nop>>pr;
    book b(t,pr,nop);
    cout<<"Enter Title,Price & Playtime(in minutes) of a
           Tape:\n";
    gets(t);
    cin>>pr>>pt;
    tape T(t,pr,pt);
    m=&b;
    clrscr();
    cout<<"\nBook Information";
    m->display();
    m=&T;
    cout<<"\n\nTape Information";
    m->display();
    getch();
}

```

/* 15. Program to create a database using concepts of files for a student including the fields: Student-name, Student's Register No, Student's Attendance (overall % of attendance) and enter data for n students and output the same in proper format*/

```

#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<iomanip.h>
#include<stdio.h>

```

```
class student
{
public:
int regno;
char name[15];
float attendance;
};

void line(int n)
{
for(int i=0;i<=n;i++)
cout<<"_";
}

void main()
{
student stu;
int n;
ofstream inputfile;
inputfile.open("stud.dat",ios::out/ios::binary);
clrscr();
cout<<"Enter number of students:";
cin>>n;
cout<<"Enter the student details for "<<n<<" students"<<endl;
for(int i=1;i<=n;i++)
{
clrscr();
fflush(stdin);
cout<<"Record no:"<<i<<endl;
cout<<endl<<"Regno:";
cin>>stu.regno;
cout<<endl<<"Name:";
cin>>stu.name;
cout<<endl<<"Attendance:";
cin>>stu.attendance;
```

```

inputfile.write((char*)&stu,sizeof(stu));
}
inputfile.close();
ifstream outputfile;
outputfile.open("stud.dat",ios::in/ios::binary);
clrscr();
int row = 5;
gotoxy(7, row++);
line(50);
gotoxy(7,row++);
cout<<setw(40)<<"Student Attendance Report"<<endl;
gotoxy(7,row++);
cout<<setw(9)<<"Reg.No"<<setw(15)<<"Name"<<setw(15)<<"Attendance";
gotoxy(7,row++);
line(45);
fflush(stdout);
while(outputfile.read((char*)&stu,sizeof(stu)))
{
gotoxy(7,row++);
cout<<setw(5)<<stu.regno;
cout<<setw(20)<<stu.name;
cout<<setw(10)<<stu.attendance;
}
outputfile.close();
gotoxy(7,row++);
line(45);
getch();
}

/* 16.Program to accessing a particular record in an employee
file */

#include<iostream.h>
#include<fstream.h>
#include<conio.h>

```

```

#include<iomanip.h>
#include<stdio.h>
class employee
{
private:
    int empno;
    char name[15];
    long int salary;
public:
    void getdata()
    {
        fflush(stdin);
        cout<<"Enter employee number,name and salary:\n";
        cin>>empno>>name>>salary;
    }
    void putdata()
    {
        cout<<setw(5)<<empno<<setw(10)<<name<<setw(10)
            <<salary<<endl;
    }
};

void main()
{
    fstream f1;
    employee emp;
    char ch;
    int n;
    clrscr();
    f1.open("emp.dat",ios::out);

    do
    {
        emp.getdata();
        f1.write((char*)&emp,sizeof(emp));
        cout<<"\n Do you want to continue(y/n) ?";
    }
}

```

```
cin>>ch;
}
while(ch=='y' || ch=='Y');

f1.close();
f1.open("emp.dat",ios::in);
clrscr();
cout<<"\n Employee details \n\n";
cout<<setw(5)<<"Empno"<<setw(10)<<"Name"<<setw(10)<<"Salary"<<
    endl;
while(f1.read((char*) &emp,sizeof(emp)))
emp.putdata();
cout<<"\n Enter the record number you want to view:";
cin>>n;
f1.close();
f1.open("emp.dat",ios::in);
int p = (n-1)*sizeof(emp);
f1.seekg(p);
f1.read((char*) &emp,sizeof(emp));
emp.putdata();
f1.close();
getch();
}
```



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

The Motto of the University

(SEWA)

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY



Bachelor of Computer Applications (BCA)

Course : Computer Graphics Lab

Course Code: BCA-4-04P

ADDRESS: C/28, THE LOWER MALL, PATIALA-147001

WEBSITE: www.psou.ac.in



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PROGRAMME COORDINATOR :

Dr. Monika Pathak

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala

PROGRAMME CO-COORDINATOR & COURSE COORDINATOR :

Dr. Gaurav Dhiman

Assistant Professor, School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University, Patiala



JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY PATIALA
(Established by Act No.19 of 2019 of Legislature of the State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in Decembas 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open Universit of the State, entrusted with the responsibility of making higher education accessible to all especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The Learner Support Centres/Study Centres are located in the Government and Government aided colleges of Punjab, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. G. S. Batra,
Dean Academic Affairs

Bachelor of Computer Applications (BCA)
BCA-4-04P: Computer Graphics Lab

Total Marks: 50
External Marks: 35
Internal Marks: 15
Credits: 2
Pass Percentage: 40%

Course: Computer Graphics Lab	
Course Code: BCA-4-04P	
Course Outcomes (COs) After the completion of this course, the students will be able to:	
CO1	Implement the basic concepts of computer graphics.
CO2	Design & Implement scan conversion problems using Python Programming
CO3	Apply clipping and filling techniques for modifying an object.
CO4	Understand the concepts of different type of geometric transformation of objects in 2D and 3D.
CO5	Understand the practical implementation of modeling, rendering, viewing of objects in 2D.

Detailed List of Programs:

Program No.	Name of Program
P1	Write a program to draw basic geometric shapes (lines, circles, rectangles) using a graphics library.
P2	Implement a program that allows the user to interactively draw and manipulate shapes on a canvas.
P3	WAP that demonstrates 2D transformations (translation, rotation, scaling) on a set of objects.
P4	Extend the program to include 3D transformations and demonstrate their effects.

P5	Implement line-drawing algorithms.
P6	Implement circle-drawing algorithms.
P7	Develop a program for polygon filling using scanline or other suitable algorithms.
P8	Create a program that renders objects with different shading models.
P9	Explore the impact of lighting models on the visual appearance of 3D objects in a scene.
P10	Implement texture mapping on a 3D model, and observe the changes in the rendered output.

Bachelor of Computer Applications (BCA)

Computer Graphics Lab

1. Write a program to draw basic geometric shapes (lines, circles, rectangles) using a graphics library.

```
#include <SDL2/SDL.h>
#include <stdio.h>

void drawShapes(SDL_Renderer *renderer) {
    // Draw a line
    SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255); // Red color
    SDL_RenderDrawLine(renderer, 100, 100, 200, 200);

    // Draw a rectangle
    SDL_Rect rect = {300, 100, 200, 150};
    SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255); // Green color
    SDL_RenderDrawRect(renderer, &rect);

    // Draw a filled rectangle
    SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255); // Blue color
    SDL_RenderFillRect(renderer, &rect);

    // Draw a circle
    int centerX = 500, centerY = 300, radius = 100;
    SDL_SetRenderDrawColor(renderer, 255, 255, 0, 255); // Yellow color
    for (int w = 0; w < radius * 2; w++) {
        for (int h = 0; h < radius * 2; h++) {
            int dx = radius - w; // horizontal offset
            int dy = radius - h; // vertical offset
            if ((dx * dx + dy * dy) <= (radius * radius)) {
                SDL_RenderDrawPoint(renderer, centerX + dx, centerY + dy);
            }
        }
    }
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("Basic Shapes",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
    SDL_RENDERER_ACCELERATED);
```

```

SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // White color
SDL_RenderClear(renderer);

drawShapes(renderer);

SDL_RenderPresent(renderer);

SDL_Delay(5000); // Pause for 5 seconds

SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();

return 0;
}

```

2. Implement a program that allows the user to interactively draw and manipulate shapes on a canvas.

```

#include <SDL2/SDL.h>
#include <stdio.h>

void handleMouseEvents(SDL_Event *event, SDL_Renderer *renderer, int *drawing,
SDL_Rect *rect) {
    switch (event->type) {
        case SDL_MOUSEBUTTONDOWN:
            if (event->button.button == SDL_BUTTON_LEFT) {
                rect->x = event->button.x;
                rect->y = event->button.y;
                *drawing = 1;
            }
            break;
        case SDL_MOUSEMOTION:
            if (*drawing) {
                rect->w = event->motion.x - rect->x;
                rect->h = event->motion.y - rect->y;
            }
            break;
        case SDL_MOUSEBUTTONUP:
            if (event->button.button == SDL_BUTTON_LEFT) {
                *drawing = 0;
            }
            break;
    }
}

```

```

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("Interactive Drawing",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
    SDL_RENDERER_ACCELERATED);

    int quit = 0;
    SDL_Event event;
    int drawing = 0;
    SDL_Rect rect = {0, 0, 0, 0};

    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
            handleMouseEvents(&event, renderer, &drawing, &rect);
        }

        SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // White color
        SDL_RenderClear(renderer);

        if (drawing) {
            SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255); // Blue color
            SDL_RenderDrawRect(renderer, &rect);
        }

        SDL_RenderPresent(renderer);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}

```

3. WAP that demonstrates 2D transformations (translation, rotation, scaling) on a set of objects.

```

#include <SDL2/SDL.h>
#include <stdio.h>

```

```

#include <math.h>

void drawRect(SDL_Renderer *renderer, SDL_Rect *rect) {
    SDL_RenderDrawRect(renderer, rect);
}

void translateRect(SDL_Rect *rect, int dx, int dy) {
    rect->x += dx;
    rect->y += dy;
}

void rotateRect(SDL_Renderer *renderer, SDL_Rect *rect, double angle) {
    SDL_Point center = {rect->x + rect->w / 2, rect->y + rect->h / 2};
    SDL_RenderDrawRect(renderer, rect);
    SDL_RenderCopyEx(renderer, NULL, rect, NULL, angle, &center,
SDL_FLIP_NONE);
}

void scaleRect(SDL_Rect *rect, double sx, double sy) {
    rect->w *= sx;
    rect->h *= sy;
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("2D Transformations",
SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
SDL_RENDERER_ACCELERATED);

    SDL_Rect rect = {200, 150, 100, 50};
    int quit = 0;
    SDL_Event event;

    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
            if (event.type == SDL_KEYDOWN) {
                switch (event.key.keysym.sym) {
                    case SDLK_RIGHT:
                        translateRect(&rect, 10, 0);
                        break;

```

```

        case SDLK_LEFT:
            translateRect(&rect, -10, 0);
            break;
        case SDLK_UP:
            translateRect(&rect, 0, -10);
            break;
        case SDLK_DOWN:
            translateRect(&rect, 0, 10);
            break;
        case SDLK_r:
            rotateRect(renderer, &rect, 45);
            break;
        case SDLK_s:
            scaleRect(&rect, 1.5, 1.5);
            break;
    }
}
}

SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // White color
SDL_RenderClear(renderer);

SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255); // Blue color
drawRect(renderer, &rect);

SDL_RenderPresent(renderer);
}

SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();

return 0;
}

```

4. Extend the program to include 3D transformations and demonstrate their effects.

```

#include <SDL2/SDL.h>
#include <GL/glew.h>
#include <GL/gl.h>
#include <GL/glu.h>

void initOpenGL() {
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);

```

```
glLoadIdentity();
gluPerspective(45.0, 4.0/3.0, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
```

```
void renderCube() {
    glBegin(GL_QUADS);

    glColor3f(1.0, 0.0, 0.0); // Red
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);

    glColor3f(0.0, 1.0, 0.0); // Green
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);

    glColor3f(0.0, 0.0, 1.0); // Blue
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);

    glColor3f(1.0, 1.0, 0.0); // Yellow
    glVertex3f(1.0, -1.0, -1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(1.0, 1.0, -1.0);

    glColor3f(0.0, 1.0, 1.0); // Cyan
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, -1.0, 1.0);

    glColor3f(1.0, 0.0, 1.0); // Magenta
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, -1.0);

    glEnd();
}
```



```

}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window *window = SDL_CreateWindow("3D Transformations",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_OPENGL);
    SDL_GLContext glContext = SDL_GL_CreateContext(window);
    glewInit();

    initOpenGL();

    int quit = 0;
    SDL_Event event;

    float angle = 0.0f;

    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
        }

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        glPushMatrix();
        glTranslatef(0.0f, 0.0f, -5.0f);
        glRotatef(angle, 1.0f, 1.0f, 1.0f);
        renderCube();
        glPopMatrix();

        SDL_GL_SwapWindow(window);
        angle += 0.5f;
    }

    SDL_GL_DeleteContext(glContext);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}

```

5. Implement line-drawing algorithms.

```

#include <SDL2/SDL.h>
#include <stdio.h>

void drawLine(SDL_Renderer *renderer, int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    int err = dx - dy;

    while (1) {
        SDL_RenderDrawPoint(renderer, x1, y1);
        if (x1 == x2 && y1 == y2) break;
        int e2 = err * 2;
        if (e2 > -dy) {
            err -= dy;
            x1 += sx;
        }
        if (e2 < dx) {
            err += dx;
            y1 += sy;
        }
    }
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("Line Drawing",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
    SDL_RENDERER_ACCELERATED);

    SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // White color
    SDL_RenderClear(renderer);

    SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255); // Black color
    drawLine(renderer, 100, 100, 700, 500);

    SDL_RenderPresent(renderer);

    SDL_Delay(5000); // Pause for 5 seconds

    SDL_DestroyRenderer(renderer);
}

```

```

    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}

```

6. Implement circle-drawing algorithms.

```

#include <SDL2/SDL.h>
#include <stdio.h>

void drawCircle(SDL_Renderer *renderer, int centerX, int centerY, int radius) {
    int x = radius;
    int y = 0;
    int radiusError = 1 - x;

    while (x >= y) {
        SDL_RenderDrawPoint(renderer, centerX + x, centerY + y);
        SDL_RenderDrawPoint(renderer, centerX - x, centerY + y);
        SDL_RenderDrawPoint(renderer, centerX + x, centerY - y);
        SDL_RenderDrawPoint(renderer, centerX - x, centerY - y);
        SDL_RenderDrawPoint(renderer, centerX + y, centerY + x);
        SDL_RenderDrawPoint(renderer, centerX - y, centerY + x);
        SDL_RenderDrawPoint(renderer, centerX + y, centerY - x);
        SDL_RenderDrawPoint(renderer, centerX - y, centerY - x);
        y++;
        if (radiusError < 0) {
            radiusError += 2 * y + 1;
        } else {
            x--;
            radiusError += 2 * (y - x + 1);
        }
    }
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("Circle Drawing",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
    SDL_RENDERER_ACCELERATED);

    SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255); // White color
    SDL_RenderClear(renderer);

```

```

SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255); // Black color
drawCircle(renderer, 400, 300, 100);

SDL_RenderPresent(renderer);

SDL_Delay(5000); // Pause for 5 seconds

SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();

return 0;
}

```

7. Develop a program for polygon filling using scanline or other suitable algorithms.

```

#include <SDL2/SDL.h>
#include <stdio.h>

typedef struct {
    int x;
    int y;
} Point;

void drawPolygon(SDL_Renderer *renderer, Point *points, int n) {
    for (int i = 0; i < n; i++) {
        SDL_RenderDrawLine(renderer, points[i].x, points[i].y, points[(i + 1) % n].x,
points[(i + 1) % n].y);
    }
}

void fillPolygon(SDL_Renderer *renderer, Point *points, int n) {
    int minY = points[0].y, maxY = points[0].y;

    // Find the min and max Y
    for (int i = 1; i < n; i++) {
        if (points[i].y < minY) minY = points[i].y;
        if (points[i].y > maxY) maxY = points[i].y;
    }

    for (int y = minY; y <= maxY; y++) {
        int intersections[64], count = 0;

        for (int i = 0; i < n; i++) {

```

```

        int j = (i + 1) % n;
        if ((points[i].y <= y && points[j].y > y) || (points[j].y <= y && points[i].y >
y)) {
            float x = points[i].x + (y - points[i].y) * (points[j].x - points[i].x) /
(float)(points[j].y - points[i].y);
            intersections[count++] = (int)x;
        }
    }

    // Sort intersections
    for (int i = 0; i < count - 1; i++) {
        for (int j = i + 1; j < count; j++) {
            if (intersections[i] > intersections[j]) {
                int temp = intersections[i];
                intersections[i] = intersections[j];
                intersections[j] = temp;
            }
        }
    }

    for (int i = 0; i < count; i += 2) {
        SDL_RenderDrawLine(renderer, intersections[i], y, intersections[i + 1], y);
    }
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);

    SDL_Window *window = SDL_CreateWindow("Polygon Filling",
SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
SDL_WINDOW_SHOWN);
    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
SDL_RENDERER_ACCELERATED);

    SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
    SDL_RenderClear(renderer);

    SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
    Point points[] = {{100, 100}, {200, 50}, {300, 100}, {350, 200}, {250, 250}, {150,
200}};
    int n = sizeof(points) / sizeof(points[0]);
    fillPolygon(renderer, points, n);
    drawPolygon(renderer, points, n);

    SDL_RenderPresent(renderer);
}

```

```

    SDL_Delay(5000);

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}

```

8. Create a program that renders objects with different shading models.

```

#include <SDL2/SDL.h>
#include <GL/glew.h>
#include <GL/gl.h>
#include <GL/glu.h>

void initOpenGL() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

    GLfloat lightPos[] = {0.0f, 0.0f, 2.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 4.0 / 3.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void drawCube() {
    glBegin(GL_QUADS);

    // Front face
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, 1.0);

    // Back face
    glNormal3f(0.0, 0.0, -1.0);

```

```

    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);

    // Top face
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, -1.0);

    // Bottom face
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);

    // Right face
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(1.0, -1.0, -1.0);
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);

    // Left face
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, -1.0);

    glEnd();
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window *window = SDL_CreateWindow("Shading Models",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_OPENGL);
    SDL_GLContext glContext = SDL_GL_CreateContext(window);
    glewInit();

    initOpenGL();

```

```

int quit = 0;
SDL_Event event;
float angle = 0.0f;
int flatShading = 1;

while (!quit) {
    while (SDL_PollEvent(&event) != 0) {
        if (event.type == SDL_QUIT) {
            quit = 1;
        }
        if (event.type == SDL_KEYDOWN) {
            if (event.key.keysym.sym == SDLK_s) {
                flatShading = !flatShading;
                glShadeModel(flatShading ? GL_FLAT : GL_SMOOTH);
            }
        }
    }
}

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glPushMatrix();
glTranslatef(0.0f, 0.0f, -5.0f);
glRotatef(angle, 1.0f, 1.0f, 1.0f);
drawCube();
glPopMatrix();

SDL_GL_SwapWindow(window);
angle += 0.5f;
}

SDL_GL_DeleteContext(glContext);
SDL_DestroyWindow(window);
SDL_Quit();

return 0;
}

```

9. Explore the impact of lighting models on the visual appearance of 3D objects in a scene.

```

#include <SDL2/SDL.h>
#include <GL/glew.h>
#include <GL/gl.h>
#include <GL/glu.h>

```



```

void initOpenGL() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

    GLfloat lightPos[] = {0.0f, 0.0f, 2.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 4.0 / 3.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void drawCube() {
    glBegin(GL_QUADS);

    // Front face
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, 1.0);

    // Back face
    glNormal3f(0.0, 0.0, -1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);

    // Top face
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, -1.0);

    // Bottom face
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);

```

```

glVertex3f(1.0, -1.0, 1.0);
glVertex3f(-1.0, -1.0, 1.0);

// Right face
glNormal3f(1.0, 0.0, 0.0);
glVertex3f(1.0, -1.0, -1.0);
glVertex3f(1.0, 1.0, -1.0);
glVertex3f(1.0, 1.0, 1.0);
glVertex3f(1.0, -1.0, 1.0);

// Left face
glNormal3f(-1.0, 0.0, 0.0);
glVertex3f(-1.0, -1.0, -1.0);
glVertex3f(-1.0, -1.0, 1.0);
glVertex3f(-1.0, 1.0, 1.0);
glVertex3f(-1.0, 1.0, -1.0);

glEnd();
}

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window *window = SDL_CreateWindow("Lighting Models",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_OPENGL);
    SDL_GLContext glContext = SDL_GL_CreateContext(window);
    glewInit();

    initOpenGL();

    int quit = 0;
    SDL_Event event;
    float angle = 0.0f;
    int lightingModel = 0;

    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
            if (event.type == SDL_KEYDOWN) {
                if (event.key.keysym.sym == SDLK_l) {
                    lightingModel = (lightingModel + 1) % 3;

                    if (lightingModel == 0) {

```

```

        glLightfv(GL_LIGHT0, GL_DIFFUSE, (GLfloat[]){1.0, 1.0, 1.0,
1.0});
        glLightfv(GL_LIGHT0, GL_SPECULAR, (GLfloat[]){0.0, 0.0, 0.0,
1.0});
    } else if (lightingModel == 1) {
        glLightfv(GL_LIGHT0, GL_DIFFUSE, (GLfloat[]){1.0, 0.0, 0.0,
1.0});
        glLightfv(GL_LIGHT0, GL_SPECULAR, (GLfloat[]){1.0, 1.0, 1.0,
1.0});
    } else if (lightingModel == 2) {
        glLightfv(GL_LIGHT0, GL_DIFFUSE, (GLfloat[]){0.0, 0.0, 1.0,
1.0});
        glLightfv(GL_LIGHT0, GL_SPECULAR, (GLfloat[]){1.0, 1.0, 1.0,
1.0});
    }
}
}

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glPushMatrix();
glTranslatef(0.0f, 0.0f, -5.0f);
glRotatef(angle, 1.0f, 1.0f, 1.0f);
drawCube();
glPopMatrix();

SDL_GL_SwapWindow(window);
angle += 0.5f;
}

SDL_GL_DeleteContext(glContext);
SDL_DestroyWindow(window);
SDL_Quit();

return 0;
}

```

10. Implement texture mapping on a 3D model, and observe the changes in the rendered output.

```

#include <SDL2/SDL.h>
#include <GL/glew.h>
#include <GL/gl.h>

```

```

#include <GL/glu.h>
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"

GLuint loadTexture(const char *filename) {
    GLuint texture;
    int width, height, nrChannels;
    unsigned char *data = stbi_load(filename, &width, &height, &nrChannels, 0);

    if (data) {
        glGenTextures(1, &texture);
        glBindTexture(GL_TEXTURE_2D, texture);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, nrChannels
== 4 ? GL_RGBA : GL_RGB, GL_UNSIGNED_BYTE, data);
        glGenerateMipmap(GL_TEXTURE_2D);
        stbi_image_free(data);
    } else {
        printf("Failed to load texture\n");
    }

    return texture;
}

void initOpenGL(GLuint *texture) {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

    GLfloat lightPos[] = {0.0f, 0.0f, 2.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    *texture = loadTexture("texture.jpg");
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, *texture);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 4.0 / 3.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void drawTexturedCube() {
    glBegin(GL_QUADS);

```

```

// Front face
glNormal3f(0.0, 0.0, 1.0);
glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, 1.0);
glTexCoord2f(1.0, 0.0); glVertex3f(1.0, -1.0, 1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, 1.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, 1.0);

// Back face
glNormal3f(0.0, 0.0, -1.0);
glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, -1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, -1.0);
glTexCoord2f(1.0, 0.0); glVertex3f(1.0, -1.0, -1.0);

// Top face
glNormal3f(0.0, 1.0, 0.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, -1.0);
glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, 1.0, 1.0);
glTexCoord2f(1.0, 0.0); glVertex3f(1.0, 1.0, 1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, -1.0);

// Bottom face
glNormal3f(0.0, -1.0, 0.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, -1.0, -1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(1.0, -1.0, -1.0);
glTexCoord2f(1.0, 0.0); glVertex3f(1.0, -1.0, 1.0);
glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, 1.0);

// Right face
glNormal3f(1.0, 0.0, 0.0);
glTexCoord2f(1.0, 0.0); glVertex3f(1.0, -1.0, -1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, -1.0);
glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 1.0);
glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 1.0);

// Left face
glNormal3f(-1.0, 0.0, 0.0);
glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, 1.0);
glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, 1.0, 1.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, -1.0);

glEnd();
}

```

```

int main(int argc, char *argv[]) {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window *window = SDL_CreateWindow("Texture Mapping",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_OPENGL);
    SDL_GLContext glContext = SDL_GL_CreateContext(window);
    glewInit();

    GLuint texture;
    initOpenGL(&texture);

    int quit = 0;
    SDL_Event event;
    float angle = 0.0f;

    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
        }

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        glPushMatrix();
        glTranslatef(0.0f, 0.0f, -5.0f);
        glRotatef(angle, 1.0f, 1.0f, 1.0f);
        drawTexturedCube();
        glPopMatrix();

        SDL_GL_SwapWindow(window);
        angle += 0.5f;
    }

    glDeleteTextures(1, &texture);
    SDL_GL_DeleteContext(glContext);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}

```