UNIT 1 SOFTWARE EVOLUTION

Structure **Page No.**

1.0 INTRODUCTION

You all must have come across computers being used at many different places – post offices, hospitals, book stores, grocery stores, universities, banks, publishing

houses, etc. You have also studied about computers and their applications in the previous block. But, have you ever wondered how the similar looking machines can behave so differently? What is it that makes them extremely useful machines for varied and unlimited purposes, unlike any other machine available to us? For example we can use a crane only to move loads – its usage is quite limited, but a computer can be used to create a document, do calculations, give presentations, book movie tickets, play movies, music or games and accomplish much more. What makes computer the versatile machines that they are?

It is the **software** that enables a computer to perform all the useful and desired functions. Different types of software help a computer to be used for multiple and varied purposes, in totally different areas of work. We will study about the software aspect of computers in detail in this unit.

1.2 OBJECTIVES

After going through this unit, you will be able to:

- define what is software;
- discuss different aspects of software evolution; and
- differentiate between types of software.

1.2 WHAT IS SOFTWARE ?

A computer system consists of two parts – hardware and software. The first part, computer hardware, refers to all the visible components of the computer system: keyboard, monitor, hard disc, printer, scanner, processing unit, memory, electrical connections, etc. It does all of the physical work a computer is known for. The second part is a set of simple and step-by-step sequence of instructions that tell the hardware what to do and how to do it. This organized set of instructions written in a defined order and to accomplish a specific task is called **computer software** or **computer program**. Hence, a computer software provides intelligence to the hardware, which otherwise is just a collection of circuits and pieces of plastic and metal.

A computer programmer writes the software that gives a computer the ability to solve any business or scientific problem.

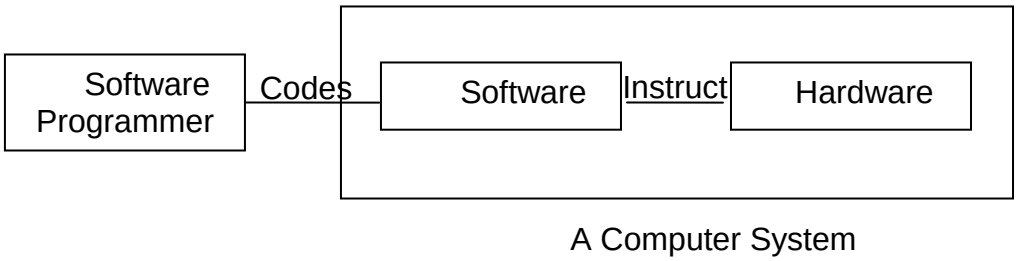

Figure 1.1: Relationship between Hardware and Software

You already know that the computer hardware is essentially a piece of complex electronics that understands only 1's and 0's – electrical "on" or "off" conditions. Hence, the instructions to perform a task must be written in a series of binary 1's and 0's. But, although this binary format, called the machine code makes perfect sense to a machine, it is incomprehensible to a computer programmer who writes the program. So a programmer codes the program in the English like programming language which is easier to understand. This program is then translated into machine code by another computer program. This translated program, called the software is eventually executed to achieve the desired goal.

You will study about types of software and its evolution in the following sections.

1.3 SOFTWARE EVOLUTION

As you know that in a computer system both hardware and software complement each other – one is of hardly any use without the other. Hence, since the very beginning of computer history, software evolution has been closely tied to the advances made in hardware. As hardware became faster, cheaper and with better capacity of storage, software became more complex and sophisticated.

Over the decades computers have been used in new areas and to solve new problems. With changing needs and improved hardware, the software has evolved in its various aspects. The software architecture, its design paradigms, programming languages, its usage, costing and licensing have all changed and evolved over the years.

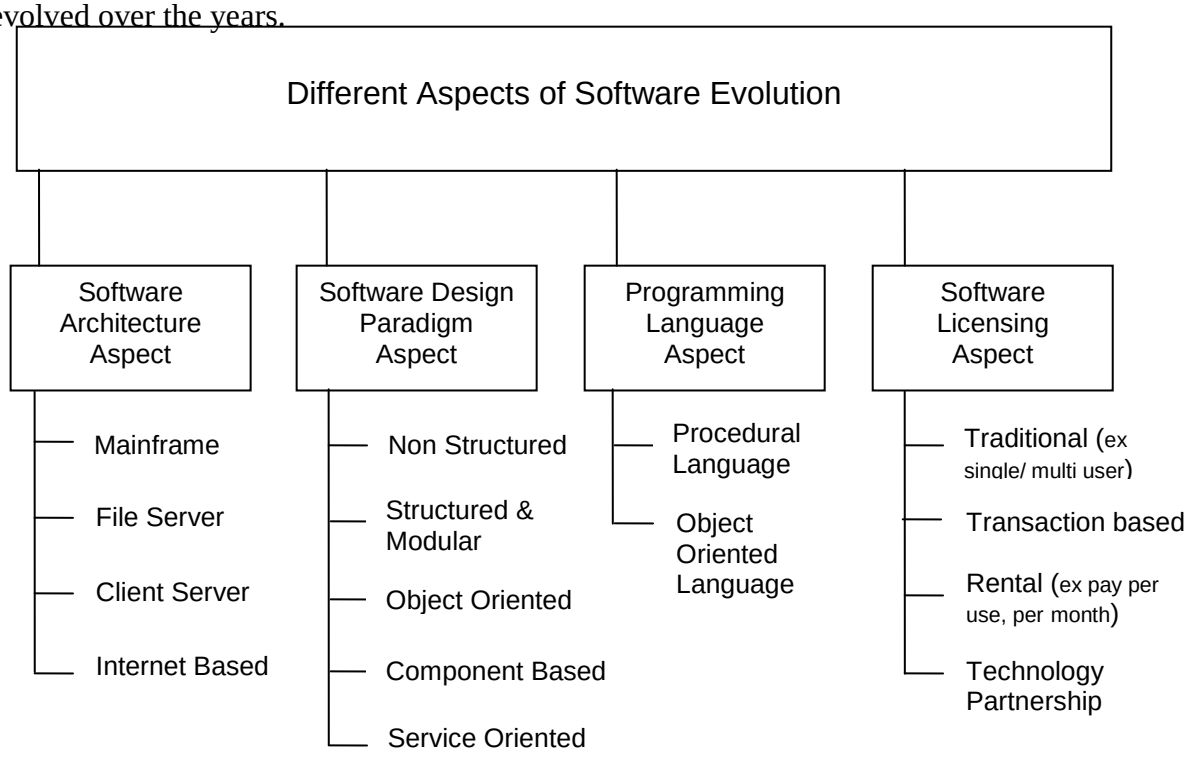

Figure 1.2: Different Aspects of Software Evolution

Software evolution with respect to its architecture, design styles, programming language and licensing will be covered in the following section.

1.3.1 Evolution of Software Architecture

The software architecture has always moved in unison with the hardware advancement.

1.3.1.1 Mainframe Architecture

Till a few decades back, all computing was controlled through the central mainframes server. Multiple users could connect to the central host through unintelligent terminals which captured the keystrokes, sent the information to the host and displayed the text output. All the processing was done by the applications residing on the main central server. Only large transaction-oriented applications were developed during that time. Business tasks such as accounts receivable, accounts payable, general ledger, credit account management and payroll that were repetitive and could be run as batch jobs were automated.

In these centralized computing models, the host provided both the data storage and processing power for the client systems. There was no support for graphical user interface or access to multiple databases from geographically dispersed sites.

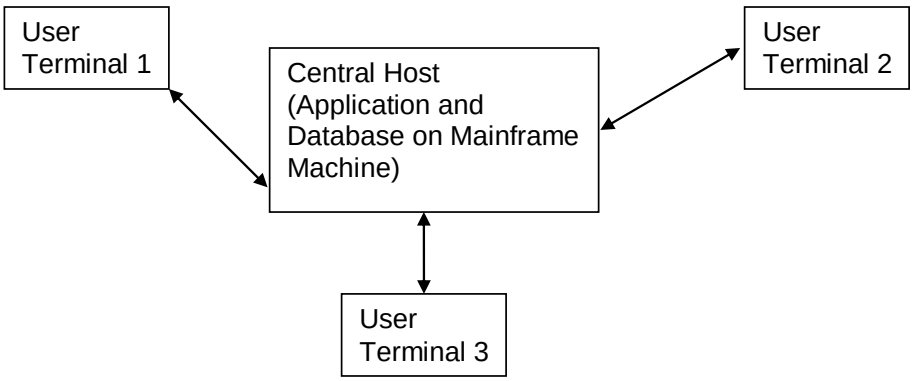

Figure 1.3: Mainframe Architecture

1.3.1.2 File Sharing Architecture

The development of microprocessor, PC and LAN transformed dumb terminals into "smart" clients. This brought a complete change in the computing environment. The client workstations or desktops, with there enhanced capabilities were now responsible for the user interface and execution of the application logic. The server provided access to computing resources like printers and large hard drives for storing the files. It downloaded the file from the shared location on the server to the client machine. The user application that worked on the data was run on the client and the file was written back to the server. The application had to be installed on each workstation that accessed the file.

In this architecture, resources could be added as and when necessary or desired. Thus, it provided a low cost entry point with flexible arrangement. The drawback

was that application logic was executed on the client and server typically provided files to store data. It worked fine as long as the volume of data transfer was low and shared usage and content update was low. As the number of online users grew, the network traffic got congested and the file sharing got strained. Taking into account the demerits of the file server architectures, the client/ server architecture made its advent.

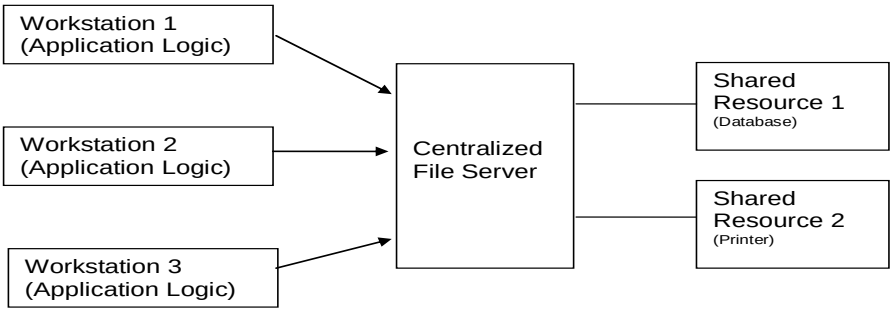

Figure 1.4: File Sharing Architecture'

1.3.1.3 Client/ Server Architecture

As the capacity and power of personal computers improved, the need to share the processing demands between the host server and the client workstation increased. This need for greater computing control and more computing value led to the evolution of client/server technology.

In client/server architecture, the tasks or workloads are partitioned as:

- server programs – providers of a resource or service
- client programs – requester of resource or service

Clients and servers may reside in the same machine or they typically reside in separate pieces of hardware and communicate over a computer network. A server machine is a host that runs one or more server programs which share their resource with clients. A client does not share any of its resources, but requests a server function or service. The server program fulfills the client request. Clients initiate a communication session with the server.
The client/ server system may be two-tiered, three-tiered or n-tiered.

Two-tiered architecture: This approach basically introduced a database server to replace the file server. The emergence of relational database management systems and graphical user interface applications led to database server which could be accessed through the GUI based client applications. Since, the clients query the database over the network and only the relevant data is supplied to the client, the network traffic is greatly reduced in comparison to the file server system.

The application or business logic in client server applications may reside on the server (fat server – thin client) or on the client (fat client – thin server). Since,

clients and server interact over the network, increases in the number of users often lead to network congestion. Also, maintenance of the application becomes difficult with more users. This lack of scalability (Ability of a system to support increased demands of work, usage or service levels almost instantly, without any change and with no significant drop in cost effectiveness or quality of service) and flexibility gave rise to 3-tiered and n-tiered architectures.

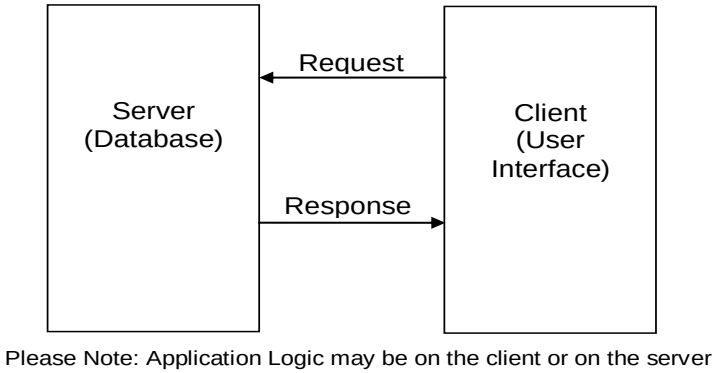

Figure 1.5: Two Tier Client Server Architecture

Three-tiered architecture: A new generation of client/server implementation takes a step further and adds a middle tier in between client and server to achieve "3-tier" architecture. The 3-tier architecture attempts to overcome some of the limitations of 2-tier schemes by separating presentation (user interface), processing (business functionality) and data into separate distinct entities. This leads to enhanced network performance and improved extensibility of business systems. Still, it has been found that three-tier methodology lacks some critical features such as reusability (Ability of a computer program to be used repeatedly with little or no modifications in many different applications) of application logic code and scalability. There may arise a situation whereby a collection of application logic code can not be reused and also they do not communicate with one another. Thus, there came a need for a viable architecture that mainly facilitates reusability of business logic as reusability phenomena has been found to reduce the cost of software development and the time to market and its quality is assured.

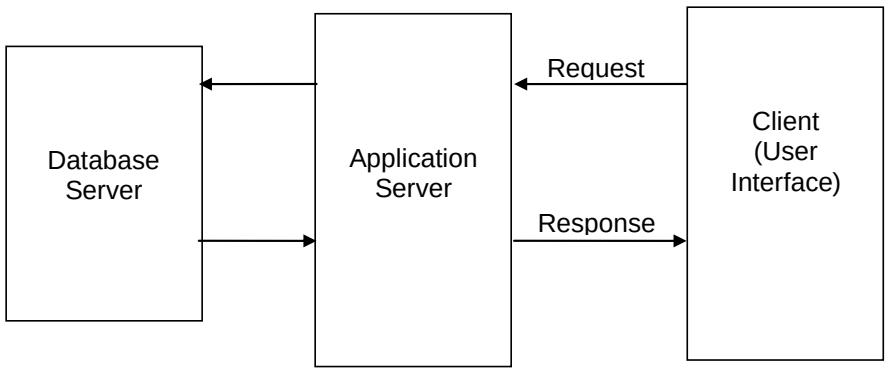

Figure 1.6: Three Tier Client Server Architecture

N-tiered architecture: The 3-tier architecture can be extended to N-tiers when the middle tier provides connections to various types of services, integrating and coupling them to the client, and to each other. Partitioning the application logic among various hosts can also create an N-tiered system. Encapsulation of distributed functionality in such a manner provides significant advantages such as reusability, and thus reliability (Ability of a computer program to perform its intended functions and operations for the specified period of time, in the specified system's environment, without experiencing any failure).

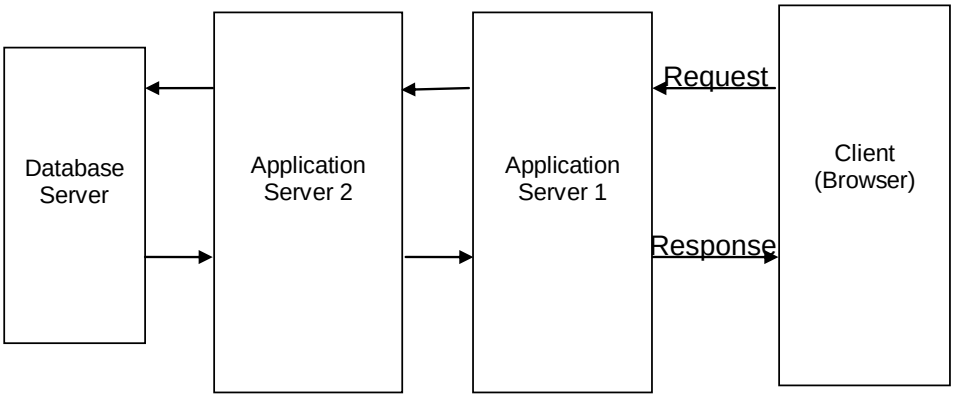

Figure 1.7 : N-Tiered Client Server Architecture

Internet-based architecture: In the late 1990's, the client/server trend was augmented by the internet. The users access the web servers through the web browsers on the client machines and over the internet. This led to very thin client based applications, which reside on corporate web servers.

The advantage of web based applications is that they do not have to be tailored to run on specific platforms. But since the web applications cannot perform client-side processing, they limit the user experience by turning the client computers into "dumb" terminals. Web mails, online transactions are examples of web applications.

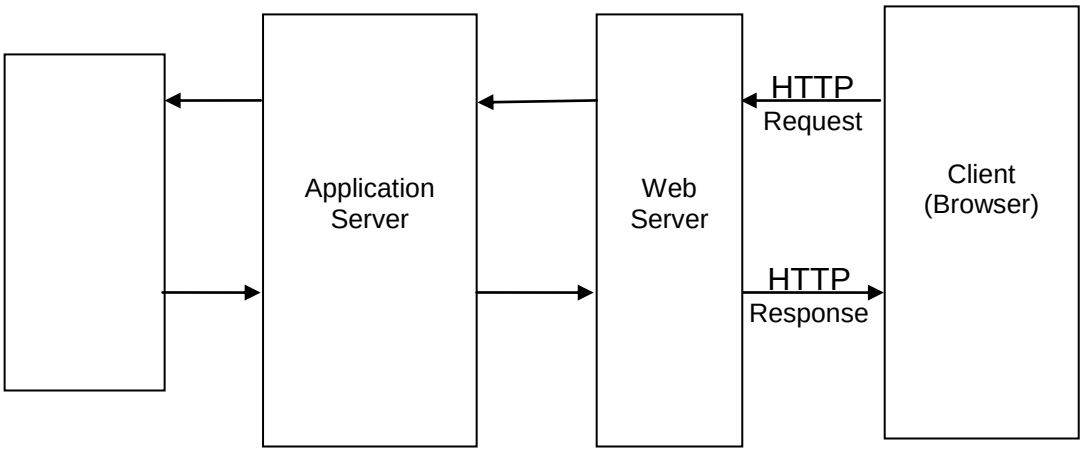

Figure 1.8 : Internet Based Architecture

1.3.1.4 Cloud Computing

As the technology has evolved from Mainframe-based large proprietary (Computer Programs that are exclusive property of their developers or publishers, and cannot be copied or distributed without complying with their licensing agreements) systems to Client-Server architecture based open systems to Open Source software based solutions, software vendor's business has also evolved over the period of time. Cloud-based software services typically mean that the consumer does not own the hardware and software, but still gets the desired service. It is an IT delivery model that offers large-scale, shared infrastructure and computing resources as a service through self-service pay-per use access. Although it leverages recently developed technology, cloud computing is a business, not a technical trend.

Here is some background for the evolution of these services. As the new software vendors tried to establish themselves in the market, they created solution differentiators which provide unique value to the consumers. An example is Salesforce.com, which from the inception offered a hosted Customer Relationship Management (CRM) solution, while its established competitors (Siebel, SAP, PeopleSoft etc) had their traditional (also called On Premise, meaning at the customer site in its dedicated environment) CRM solution. Another reason is that software vendors started targeting a niche customer segment called Small & Medium Business (SMB). SMB customers are relatively new in business, so need to establish the core IT systems in place and also have lesser financial strength, as a result are more open towards cloud-based solution.

An early example of cloud based computing is web-based emails (hotmail, yahoo, gmail etc), Chat (AOL, MSN etc). Here the required computer resources are provisioned centrally in the cloud (internet) and shared by the user pool. These days, more often, software is bundled with the required shared infrastructure to provide a solution stack to the consumer.

Key features of cloud computing are:

- **Infrastructure sharing:** Cloud computing enables dynamic sharing of resources so that demands can be met cost effectively.
- **Scalability:** To handle ever increasing workload demands and support the entire enterprise, cloud computing must have the flexibility to significantly scale IT resources.
- **Self service:** Cloud computing provides customers with access to IT resources through service-based offerings. The details of IT resources and their setup are transparent to the users.
- **Pay-per-use:** Because cloud resources can be added and removed according to workload demand, users pay for only what they use and are not charged when their service demands decrease.

There is another term that is associated with cloud computing:

- Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) – also referred to as On Demand software. This is a software solution delivery model where the software and the associated data are hosted centrally (in the cloud) and are accessed by the consumer through a thin client such as a web browser. Common applications for this are business applications such as – Accounting, Collaboration (Email, Messenger, Web meeting etc), Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Human Resource Management etc.

Key benefits of Cloud-based solutions are:

- Lower upfront cost to get started, lower time-to-market (as it takes less time to get a customer going on a cloud solution), allows the company to focus on the core business and not worry about hiring and constantly training its staff on the new technology etc.

- On the flip side for a Cloud-based solution, certain segment of customers such as large Banks and Financial institutions, Insurance companies may have security constraints in letting their data reside outside its premises (in their own data centers).

1.3.2 Evolution of Software Design Paradigm

As the software evolved in its complexity, architecture and use, and as the programming languages got better, styles of software programming also changed and improved.

One of the longest standing goals of software design is reusability which leads to increased reliability, accelerated development and easy maintenance. Over the years, software languages and software design paradigms which have evolved, all encourage compartmentalization of functionality to achieve this goal. Similar functionality is grouped together into small, independent and reusable units. These units can be used in any application for the purpose for which they were originally intended.

The first step towards compartmentalization was moving from line by line non structured program design to procedure-oriented program design.

1.3.2.1 Non-structured Design Paradigm

Non-structured programming is historically earliest programming paradigm. A non structured program usually consists of sequentially ordered statements, usually one in each line. The lines are usually numbered or labeled to allow the flow of execution to jump to any line in program. There is no concept of procedures in non structured program; hence there are no independent reusable

units in this programming paradigm. The program flow in non-structured programming would be as follows:

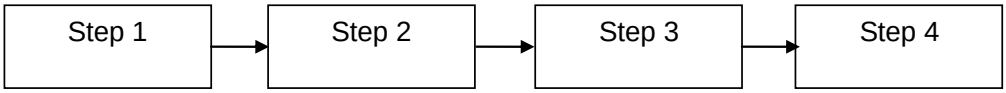

Figure 1.9: Program flow in Sequential Non Structured Programming

Example of code in non structured programming is:

```
        INTEGER i
        i=12
1       GOTO 10
2       CONTINUE
        i = i - 1
        IF (i = 0) GOTO 99
10      PRINT*, "Line 10"
        GOTO 2
99      CONTINUE
```

The above fragment of code simulates a loop using GOTO statement for transfer of control. Note that the lines are labeled/ numbered so that they can be used with GOTO. The program executes the statements sequentially. The code simulates a loop to decrease the value of variable i by 1 till it reaches zero.

The initial value of i is 12. Until the value of i reaches zero, it continues to print the text “Line 10”.

1.3.2.2 Structured and Modular Design Paradigm

Structured design paradigm introduced the concept of selection and repetition of statements in code execution along with the line by line execution. It allowed writing of procedures and functions. These are the terms used for a block of code that is written to perform a single task. Procedures and functions were the beginning of compartmentalization and hence reusability of program code.

Procedures and functions which were for similar purpose were grouped together to get a module. A big software application consisted of multiple modules, each performing a particular task.

Structural design allowed modules to be reused in the form of code libraries.

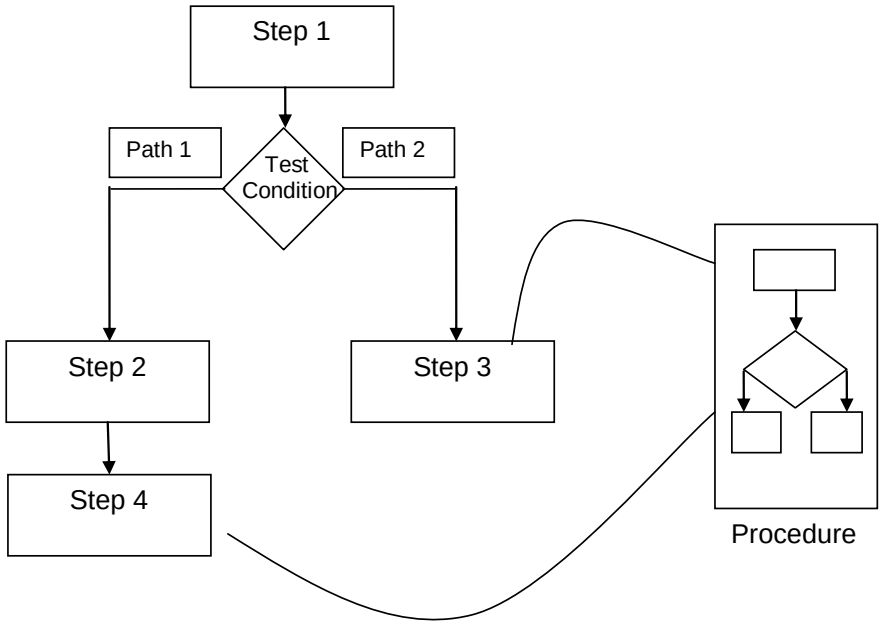

Figure 1.10 : Structured Programming with Procedure

As shown in Figure 1.10, same procedure is invoked from step 3 and step 4. There is also a selection of path to be followed. The two paths would be either steps 1,2,4 or steps 1,3.
The sample pseudo code for the above flow could be:

Steps from Figure 1.10	**Code Corresponding to Steps fromFigure 1.10**

Step 1
Test Condition – Path 1
Step 2
Step 4
Test Condition – Path 2
Step 3
Procedure
The procedure again has multiple steps.
 	
Note that in this case Step 3 and Step 4 are same – Call to the procedure PrintPrime()
	Accept user input in X
	If X is even then
	Add 1 to X
	Call procedure PrintPrime(X)
	Else
	Call procedure PrintPrime (X)
	Procedure PrintPrime (Y)
	Accept Y and Check if Y is prime number
	If Y is prime
	Display “Y is prime number”
	else
	Display “Y is not prime number”

1.3.2.3 Object Oriented Design Paradigm

The next leap forward towards compartmentalization and reusability was arrival of object oriented-design that introduced the concept of an object as an atomic unit of functionality.

Object oriented design is built on the premise that programming problems can be modeled in terms of the objects in the problem domain. In this design, any object of interest in the real world is an object in the program code. This helps to effectively model the real world and interactions of items within it. As the objects in the real world interact with each other, similarly, the objects in the program interact through their interfaces or messages which are very well defined and contained in the objects. An object exposes the interface that can be called on by other objects that need the object's functionality. Because an object is a self-contained entity and because its interface is well-defined it is highly reusable across many applications.

For example, school and student are objects in real world. They would be considered as objects in object oriented design also and they would interact with each other through messages.

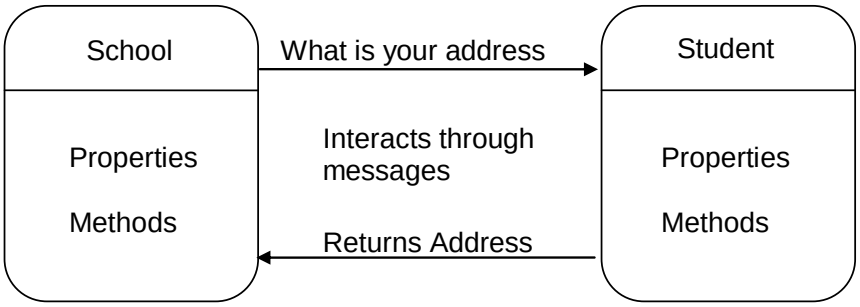

Figure 1.11: Object Oriented Programming

In the Figure 1.11 each object has its own properties and methods which constitute the interface of the object.

Encapsulation is one of the basic principles of object-oriented programming (OOP) and refers to an object's ability to conceal its data (properties) and methods. Encapsulated objects only publish the external interface so any user interacting with them only needs to understand the interface and can remain ignorant as to the internal specifications.

As the applications grew complex, code became more modular and reusable. The applications were being broken up into pieces that were distributed across many machines. Breaking applications into multiple parts and distributing them across multiple platforms presented a new set of reusability problems.

1.3.2.4 Component Based Paradigm

The concepts of Object Oriented paradigm were extended to component based programming. Component Based Development owes many concepts to object-oriented methods. It gives a more abstract view of software systems than object-oriented methods. This model prescribes that programming problems can be seen as independently deployable black boxes (components) that communicate through contracts.

The meaning of component or component based programming is intuitive: programs are broken down into primitive building blocks, which may be flexibly "plugged together" according to well-defined protocols.

The idea behind component based programming is to develop software systems by assembling a set of independently developed components off-the-shelf (COTS) in a 'plug and play' manner. For example, a Shopping cart website application may use off the shelf credit card authorization component.

Components exist at different sizes varying from single objects inside a library to whole applications. In most cases, however, components are larger entities and contain several objects.

Components are regarded as part of the starting platform for service-orientation – whereby a component is converted into a service.

1.3.2.5 Service Oriented Paradigm

Service-Oriented Programming builds on Object oriented programming, adding the premise that problems can be modeled in terms of the services that an object provides or uses.

A service is a unit of functionality defined by a set of message exchanges that are expressed using an implementation neutral grammar. It is a behaviour that can be implemented or provided by any component for use by any component based on message exchange.

A service, unlike an object, is an abstract entity whose implementation details are left largely ambiguous. The only implementation details spelled out are the messages the service exchanges or message exchanges. This ambiguity, coupled with the requirement that the messages be defined by an implementation neutral grammar make a service highly reusable and easy to integrate into a complex system.

1.3.3 Evolution of Programming Languages

As the software architecture moved from mainframes to Internet based and design paradigms evolved from non-structured to service oriented, Programming languages evolved to support the architecture and the design paradigms. As design became more and more compartmentalized so that the application could be distributed onto multiple machines, and individual components could be reused, more and more programming languages were designed to make support these ideas. For example COBOL is one language that evolved from procedural to object oriented.

1.3.3.1 Procedural Language

Procedural programming could also be called linear programming as one thing happens and then the next. Each instruction is executed in order from the top of the file to the bottom. It focuses on the idea that all algorithms are executed with functions and data that the programmer has access to and is able to change. Some languages which support procedural programming are C, FORTRAN, VB, etc.

Let us consider an example to understand how a procedural language works. You need to create forms for online inventory system for an automobile parts manufacturer. You are asked to design two separate forms: one to process information about cars and other about trucks.

For cars, we will need to record the following information:

- Color
- Engine Size
- Transmission Type
- Number of doors
- Make

For trucks, the information will be similar, but slightly different. We need:

- Color
- Engine Size
- Transmission Type
- Cab Size
- Towing Capacity
- Make

We will code as follows:

```
/*Declare the Global variables*/

Var Color
Var EngineSize
Var Transmission Type
Var Make

MainProg()
Begin
    If requested for car
        Call CarProcedure()
    If requested for Truck
        Call TruckProcedure()
End
CarProcedure()
Begin
```

```
        /*Declare the Local variables*/
        Var NumberOfDoors

    Process Car Information
End

TruckProcedure()
Begin
        /*Declare the Local variables*/
    Var CabSize
    Var TowingCapacity

    Process Truck Information
End
```

If we need to add form to process information for bus, then we need to change the MainProg() and add code for Bus Form. But if there is a change in the processing of all the vehicles, then we need to make changes to all the forms. If we need add any make specific information for cars, then we need to create multiple forms, one for each make. Also, we need to be careful about any changes to the global variables as all the forms are accessing them.

1.3.3.2 Object Oriented Language

Object Oriented Programming is more abstract than procedural programming because it looks for patterns and reusability. The same code can be loaded and executed many times to accomplish a task without having to retype it.

Before we consider above example in object oriented, let us understand few terms and concepts associated with object oriented programming. There are three main concepts that any language needs to support to be an object oriented language.

Encapsulation: is a mechanism through which a protective wrapper is created to hide the implementation details of the object and the only thing that remains externally visible is the interface of the object. (i.e.: the set of all messages the object can respond to). Encapsulation prevents code and data from being arbitrarily accessed by other code defined outside the wrapper.

Inheritance: is the process by which a new class is created using an existing class. It is a way to compartmentalize and reuse code since it allows classes to inherit commonly used state and behavior from other classes. The new classes are called the derived classed and the main class is called the parent class.

Polymorphism: Polymorphism is the characteristic of being able to assign a different meaning specifically, to allow an entity such as a variable, a function, or an object to have more than one form. It is the ability to process objects differently depending on their data types and to redefine methods for derived classes.

Following are the few terms that will help you understand object oriented programming:

- A class is a set of functions that work together to accomplish a task. It can contain or manipulate data, but it usually does so according to a pattern rather than a specific implementation. An instance of a class is considered an object.
- An object receives all of the characteristics of a class, including all of its default data and any actions that can be performed by its functions. The object is for use with specific data or to accomplish particular tasks. To make a distinction between classes and objects, it might help to think of a class as the ability to do something and the object as the execution of that ability in a distinct setting.
- A method simply refers to a function that is encased in a class.
- A parameter is a variable that is passed into a function that instructs it how to act or gives it information to process. Parameters are also sometimes called arguments.
- A property is a default set of data stored in a class. A class can have multiple properties and the properties can be changed dynamically through the methods of the class.

Smalltalk, C++, Java, C# are some of the examples of object oriented languages. Now let us see how we create classes and use them for the automobile parts inventory management system example.

```
Class Vehicle
{/*Data*/
        Var Color
        Var EngineSize
        Var TransmissionType
        Var Make
/*Methods for each data*/
    Color()
    {
        Store and update color;
    }

        EngineSize()
        {
            Store and update EngineSize;
```

```
        }

        TransmissionType()
        {
            Store and update TransmissionType;
        }

        Make()
        {
            Store and update Make;
        }
}
Class Car Inherits Vehicle
{/*Data*/
        Var NumberOfDoors

/*Methods*/
    NumberOfDoors()
        {
                Store and update NumberOfDoors;
        }
}

Class Truck Inherits Vehicle
{/*Data*/
        Var CabSize
    Var TowingCapacity

/*Methods*/
    CabSize ()
        {
                Store and update CabSize;
        }

        TowingCapacity ()
        {
                Store and update TowingCapacity;
        }
}
```

Now in this case, if we need to add form to process information for bus, then we just add one more class Bus() which is again inherited form the Vehicle class. And if we need add any make specific information for cars, then again we can add make specific classes which can be inherited from the Class Car() We need not worry about mistakenly modifying any global variables. If there is change in processing of all the vehicles, then instead of making changes at all the places, we just modify the Vehicle() class.

1.3.4 Evolution of Software Licensing

Software licensing has kept pace with the evolution of software solutions offered by the vendor or the solution provider community.

1.3.4.1 Introduction to Software Licensing

Until early 1970's, sharing of software was the accepted norm. Hardware came bundled with software products which could be freely redistributed and the access to source code allowed its improvement and modification.

In late 1960's, the situation changed after the software cost increased and manufacturers started to unbundle the software and hardware. A growing amount of software was now developed for sale. In late 1970's and early 1980's companies began imposing restrictions on programmers through copyright. They achieved financial gains by selling rights of use of software rather than giving the source code. This led to introduction of **software licensing** which governed the usage and redistribution of software. During this time most of the companies developed **proprietary software** that was actually the property of the company, came without the source code and the users basically bought the right to use it in the way specified under the license agreement.

In early 1980's the seeds for free and **open software** were sown as a deviation from the proprietary software. The open source software comes with source code and a license that allows modification and free redistribution.

We will study in the following section, about different types of licenses that evolved with software over the period of time.

1.3.4.2 Types of Software Licensing

The licensing type generally depends on whether the software is open source software, is meant for individual use or enterprise wide commercial use:

Individual License: allows you to install the software only on a single stand alone machine. It may be a **perpetual license** or **Subscription based.** Perpetual license allows you to install and use the software indefinitely. Subscription based license allows you to use the license for the specified time, after which you may renew the subscription or remove the software.

Open Source License: It grants you the right to freely modify and redistribute the software.

Commercial License: These are mostly for the large enterprises that use software for commercial purposes.. Following are the main licensing models:

- **Traditional model :** This includes single user-single license, multi users-shared license, temporary or fixed-period licenses. This has mostly been used for large proprietary mainframe applications.
- **Transaction-based model :** Here, the pricing is based on providing a committed business service, for ex, processing payroll for a global company as part of HR offering and this can be priced per employee. Larger the employee base at a given location, lower the price / employee can be. This model came into existence with the evolution of software architecture from mainframes to internet based. As mentioned before, when a company provided a particular business service, its client could access the system from anywhere over the Internet and they need not bother about maintaining the database or the software system. The service provider then charges them for each transaction/ record processed through their system.
- **Rental model :** This has come into picture as Software as a Service (SaaS) and Platform as a Service (PaaS) models have evolved over a period of time. Here, the buyer need not need make upfront investment in hardware and software, rather these come as bundled service to them. Few examples where these are prevalent are – Finance & Accounting (Core Finance), Human Resources (Core HR), Analytics (Business Intelligence/Reporting), Procurement etc. There are also scenarios where the software vendor provides subscription of a given solution (ex. Windows Azure, Salesforce.com, Siebel On Demand, Amazon Web Services etc) on a periodic (ex. monthly, annual) basis.
- **Technology Partnerships :** Such agreements provide the consumer un-limited access to vendor's technology. Such contracts are typically multi-year in nature where the consumer pays a fixed annual fee, which can be adjusted in the subsequent years based on the actual usage. For example, a large corporate customer deciding to use Oracle suite of ERP (Finance, HR), database, CRM, Business Intelligence/ Reporting solutions can get into a long-term multi-year partnership.

Check Your Progress 1

1. Compare and contrast the following:
   a. Mainframe and File Sharing architecture
   b. Client server and Distributed architecture
   c. Structured and Non Structured Programming

   ……

   ……

   ……

2. Describe the following terms:
   a. Software Reusability
   b. Software Reliability
   c. Encapsulation

……

……

……

3. What do you understand by Software-as-a-Service? How is it different from Cloud Computing?

……

……

……

4. What is pay-per-use licensing?

……

……

……

1.4 TYPES OF SOFTWARE

There is a wide variety of software available today. And there is no clear cut distinction in certain software systems. Still, most computer software can be broadly classified as:

- System software
- Programming software
- Application software

At times the categorization is vague and some software may fall into more than one categories.

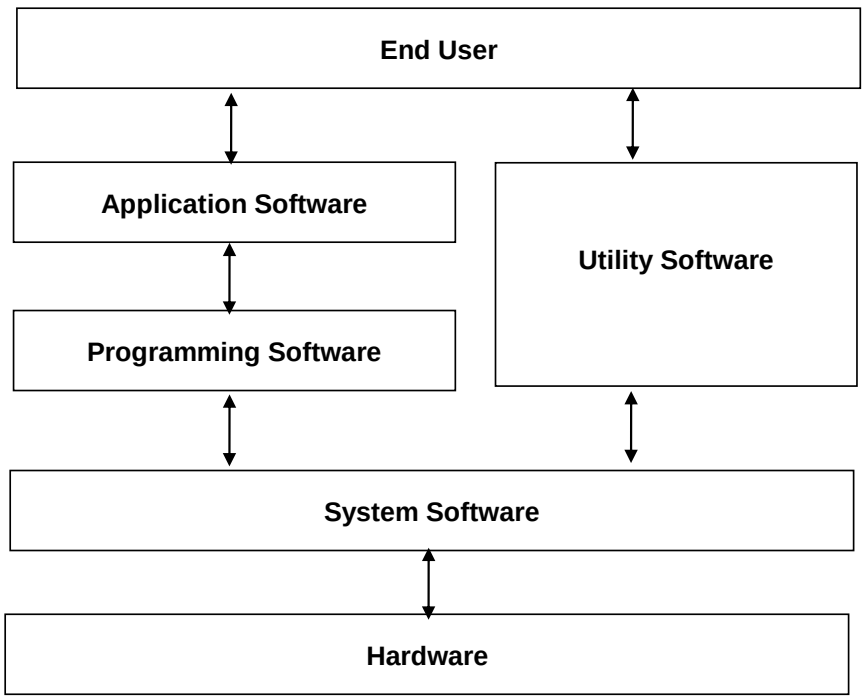

Figure 1.12: Types of Software and their interrelationship

1.4.1 System Software

System software helps run the computer hardware and system. It is designed to control the operations of a computer and coordinate all external devices like communication devices, printers, keyboards, display units, etc. It manages all the computer resources like memory and processor time in optimal and stable manner.

System software provides a useful link between user and computer. It also assists the computer in the efficient control, support, development and execution of application software. System software is essential for computer hardware to be functional and useful.

Some common types of system software are:

a) Operating Systems

Operating System is the software that manages all the computers' resources to optimize its performance provides common services for efficient execution of various application software and acts as an interpreter between the hardware, application programs and the user.

An operating system is essential for any computer to be useful to us. When a user or a program wants the hardware to do something, the request is always communicated to and processed by the operating system.

Operating systems performs basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk and controlling peripheral devices.

For large systems, the operating system has even greater responsibilities and powers.

Most operating systems perform the functions given below:

- Process Management
- Memory Management
- File Management
- Security
- Command interpretations

You will study in detail about the operating system in the following units in this block.

b) Server Programs

Server programs are dedicated computer programs that run as services and serve the needs or requests of other programs. These services may run on a dedicated hardware or on the same computer as the requesting program. Also, one on more services may run on the same computer hardware. Some common examples of different types of server programs are:

- Web server – for hosting websites.
- Print server – manage multiple print requests for multiple printers.
- File server – manages the storage and retrieval of shared computer files.
- Database server – provide database services to other computer programs.
- Mail Server – manages and transfers electronic mail messages.

c) Device Drivers

Device drivers are shared computer programs that provide an interface between the hardware devices and operating system or other higher level programs.

You need a specific software program to control each hardware device attached to the computer. It is very tedious to make any piece of hardware work. For example to write to a hard disk, you need to know the specific address available, wait till hard disk is ready to receive data and then feed it with data once it is ready. So instead of writing the same code for a device in multiple applications you share the code between applications. To ensure that the shared code is not compromised, you protect it from users and programs. Such a piece of code is called the device driver.

Device drivers are hardware dependent and operating system specific. They allow you to add and remove devices conveniently from your computer system without changing any of the applications using that device.
Common hardware components that require drivers are:

- Keyboards
- Mouse
- Printers
- graphics cards
- sound cards
- card readers
- CD/ DVD drives
- Network cards
- Image Scanners

d) Communications Software

In a networked environment, the communication software or network operating system allows computers to communicate with each other. It enables sharing and transferring of data across the network. It controls network operations and manages network security.

1.4.2 Programming Software

Programming software usually provides tools to assist a programmer in writing computer programs, and software using different programming languages in a more convenient way. It shields the application software programmer from the often complex details of the particular computer being used.

Programming Software includes the following:

a) Compilers

A compile is a program that translates the code written in a high-level programming language (called the source code) to the code in lower level language (the object code). The compiler translates each source code instruction into a set of, rather than one object code instruction. Generally, the object code is the machine language code.

When a compiler compiles a program, the source program does not get executed during the process, it only gets converted to the form that can be executed by the computer.

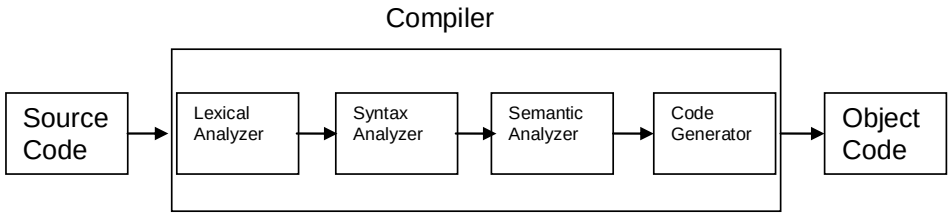

Figure 1.13: Compiler

b) Debuggers

A debugger or debugging tool is a computer program that is used to test and debug other programs (the target program).

Typically, debuggers offer functions such as running a program step by step (single-stepping) or breaking (pausing the program to examine the current state) at some event or specified instruction by means of a breakpoint, and tracking the values of some variables. Some debuggers have the ability to modify the state of the program while it is running, rather than merely to observe it. It may also be possible to continue execution at a different location in the program to bypass a crash or logical error.

c) Interpreters

Interpreter is another translation program. It takes the source code instruction, one at a time, translates and executes it.

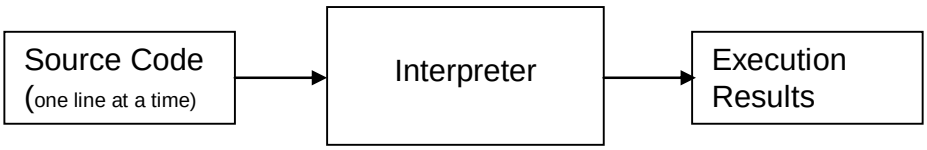

Figure 1.14: Interpreter

d) Linkers

A linker or link editor is a program that takes one or more Object file codes generated by a compiler and combine them into a single executable program.

When large software, involving many programmers is to be developed, then the modular approach is adapted. The software is divided into functional modules and separate source programs are written for each module. Each of these source files can then be compiled independent of each other to create a corresponding object file. Eventually, linker is used to combine all the object files and convert them into a final executable program.

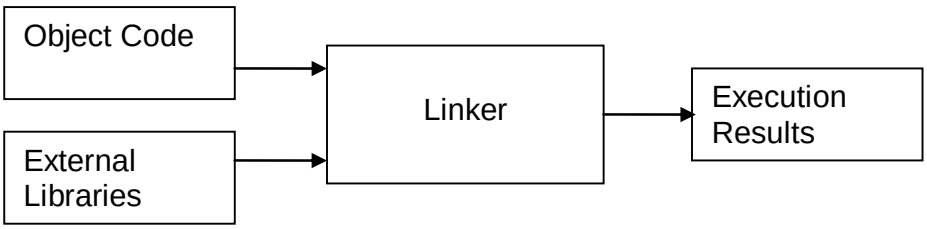

Figure 1.15: Linker

e) Text editors

A text editor is a type of program used for editing plain text files.

Many text editors for software developers include source code syntax highlighting and automatic completion to make programs easier to read and write.

Common text editors in Windows environment are Notepad and Textpad.

1.4.3 Application Software

Application software is designed and developed to accomplish one or more specific task or solve a particular problem.

Application software may be for commercial or scientific use. There is wide range of application software available for varied purposes. Some major categories of these applications include:

a) Word Processing Software can be used to create, edit, format, save, view or print any text based document like letters, memos, reports, etc. MS Word is an example of word processing software

b) Spreadsheet Software can be used to create any numeric based documents or as numeric data-analysis tool. For example it can be used to make budgets, financial statements, comparative charts, etc. MS Excel is an example of Spreadsheet software.

c) Database Software can be used to store, maintain, manipulate and organize a large set of data. For example, it can be used to maintain address, phone number directory, client directory, etc. Oracle is an example of database software.

d) Presentation Software like MS PowerPoint can be used to create and present slide show.

e) Graphics Software can be used to manage and manipulate pictures, photographs, movies, etc. Photoshop, Illustrator and MS Paint are examples of graphics software.

f) Multimedia Authoring Application can be used to create digital movies with sound, video, animation and interactive features. Mediator 9 is an example of multimedia authoring tool.

Other applications include:

- Entertainment and Education Software
- Industrial automation
- Business software like inventory management, airline reservation
- Video games
- Telecommunications
- Mathematical software
- Medical software
- Scientific software like molecular modeling, quantum chemistry software
- Image editing
- Simulation software
- Decision making software

Check Your Progress 2

1. Compare and contrast the following:
   a. System and Application Software
   b. Compiler and Linker
   c. Compiler and Interpreter

   ……

   ……

   ……

2. Give an example of each of the following:
   a. Decision Making Software
   b. Education Software
   c. Industrial Automation Software
   d. Mathematical Software
   e. Simulation Software

   ……

   ……

   ……

3. You bought a new printer. You attached it to the computer and plugged to the power, but it still does not work. What do you think must have happened and how can you resolve the issue?

   ……

   ……

   ……

4. List which software will be required to perform the following actions:
   a. You have write code in C++. What software you will use to write the code in?
   b. You have attached a new scanner to your machine to scan your photographs. What software you will use to get it working?
   c. You have bought a new PC. What is the first piece of software that is needed to be installed for it to be useful so that other software could be added?
   d. You have created a student registration system. What will you use to store the students data.

1.5 UTILITY SOFTWARE

Utility programs help manage, maintain and control computer resources. These programs are available to help you with the day-to-day chores associated with personal computing and to keep your system running at peak performance.

Some of utility programs are discussed below:

Anti-virus software

Computer viruses are software programs that are deliberately designed to interfere with computer operation; record, corrupt, or delete data; or spread themselves to other computers and throughout the Internet. Virus Scanning Software are utility programs designed to protect your computer from computer viruses, worms and trojan horses.

Historically, computer viruses were associated with self-reproducing executable programs that manipulated or even destroyed data on infected computers. They were known to spread by infected floppy disks, network or other hardware media. With the advent of internet, the viruses spread online as well. They can also spread through powerful macros used in word processor applications, like MS Word, or email programs where viruses are embedded in the email body itself and reproduce when the message is just opened or previewed.

To help prevent the most current viruses, you must update your antivirus software regularly. You can set up most types of antivirus software to update automatically.

Most anti-virus programs use one of the following techniques to identify viruses:

1. **Signature based detection**: This is the most common method. It compares the contents of the infected file to a known pattern of data. Because viruses can embed themselves in existing files, the entire file is searched.
2. **Heuristic-based detection**: This method is primarily used to identify unknown viruses by looking for malicious code or variations of such code.
3. **File emulation**: This is another heuristic approach in which the infected program is run in a virtual environment and the actions it performs are recorded. The actions are analyzed to check for any malicious actions and carry out disinfection actions accordingly.

No matter how useful antivirus software can be, these can sometimes have some drawbacks.

- Antivirus software can impair a computer's performance. Active anti-virus programs can cause conflicts with other programs.
- A "false positive" is when antivirus software identifies a non-malicious file as a virus. When this happens, it can cause serious problems. For example, if an antivirus program is configured to immediately delete infected files, a false positive in a essential file can render the operating system or some applications unusable.
- Most popular anti-virus programs are not very effective against new viruses. The reason for this is that the virus designers test their new viruses on the major anti-virus applications to make sure that they are not detected before releasing them into the market.

- Some apparent antivirus programs are actually malware being sold as legitimate software, such as Win Fixer and MS Antivirus.
- Some commercial antivirus software agreements include a clause that the subscription will be automatically renewed. For example, McAfee requires users to unsubscribe at least 60 days before the expiration of the present subscription. Norton Antivirus also renews subscriptions automatically by default.
- Finally, antivirus software generally runs at the highly trusted kernel level of the operating system, creating a potential avenue of attack

Despite the drawbacks, anti-virus software have become a necessity these days. A number of popular anti-virus programs include those by Kaspersky, Symantec, McAfee, and Norton. The cost of the program increases with the increase in the number of virus detection and removal features and ease they offer.

Backup utilities

Backup refers to making copies of data so that these additional copies may be used to restore the original after a data loss event. All types of data could be backed up like pictures, word documents, files, executables or an entire database.

The main purpose is to recover data in the event of data loss or data getting corrupt. Other purpose could be to recover historical data.

A number of Backup software are available that assist you in taking backup of your important data on the computer. Selecting between various back-up software is not only a based on the cost but also on the software that meeting the requirements.

A backup software could allow automated scheduling of backup in addition to just creating copy of files. The software should be easy to install and maintain. It should be intuitive and easy to use. The restoring from the back-up should be simple. Accessing restored data should be automatic, and the backup should preserve original data files and paths. A backup software that can compress data helps in storing data in lesser space. Certain software also allows securing the backed-up data with passwords and encryption. Good documentation and technical support goes a long way in ensuring help is available when needed.

Backup could be taken on variety of media including hard drive, CDs, DVDs, floppy disks etc. It could also be taken on FTP locations, tape or online servers. A number of free and proprietary back-up software are available including those from Microsoft, Symantec, Apple, IBM, and Norton.

It is important to take backup of important data regularly and also verify that it can be restored successfully.

Diagnostic programs

A diagnostic program is a program written for the purpose of locating problems with the software, hardware, or both, or a network of systems. A diagnostic program provides solutions to the user to solve issues.

In practical experience, these tools do not usually identify the exact cause of the system problem, but they often provide some information about what is in the system and how it is working. Some of these are free or are included with common operating systems at no additional charge, while others are commercial products that range from affordable to rather pricey.

Here are some common software diagnostic tools.

- **Power-On Self Test (POST) :** This isn't a separate diagnostic utility; it is in fact built into your system BIOS and it runs every time you start up your PC automatically. It is often the best indicator of system problems. Don't disable its error-reporting functions unless you really need to.
- **MEM.EXE :** This simple utility, built into Windows operating system that provides you with details about your memory configuration, as well as what is currently using your memory.
- **Microsoft Diagnostics :** Better known as "MSD.EXE", this is a small DOS utility that takes a brief inventory of the contents of your PC and shows them to you in a text-based format. This is very useful for seeing what disks are in the system, how much memory is installed, and also for checking system resource usage. It will show you what type of BIOS you are using.
- **The Windows Device Manager :** This is the most useful tool for identifying system configuration and resource usage information.
- **Norton System Information :** This utility is similar to the Microsoft Diagnostics, only more detailed in its later versions. SI shows a great deal of information about what is in the PC, going well beyond what MSD gives you, but really is still an information utility as opposed to a true diagnostic. This program is part of Symantec's Norton Utilities.
- **Microsoft ScanDisk and Norton Disk Doctor :** These programs are used to check for hard disk problems. This includes file system corruption and hard disk read errors. They should be used when hard disk problems are suspected.
- **Scandisk** is a utility provided with Windows computers. Scandisk scans your disks to see if there are any potential problems on the disk, such as bad disk areas. Since disks are magnetic media, all disks, including your hard drive can be corrupted
- **Microsoft Disk Defragmenter** software assists you in keep reorganizing your disk drives. After files are saved, deleted and resaved again, the disk can become fragmented --- available space is in small blocks located throughout

the disk. Disk defragmenters gather those free spots and put them together to enable you to continue to save your data in the most efficient manner.

- **Norton Diagnostics :** This utility is meant to go beyond the System Information program and actually perform tests on the hardware to identify problems. It includes tests of the processor and motherboard and system memory, and will identify some types of resource conflicts. In reality, it is still quite limited in terms of the numbers of problems it will find.
- **QAPlus :** QAPlus from DiagSoft is a more advanced diagnostic suite that comes in several flavours, depending on what you need to do and how you want to do it. This is a more expensive package but can give you much more detailed information about your system and help identify problem situations as well.

File view programs

File view utilities let you see the contents of a wide variety of documents even when you don't have the application on your system

A file viewer is limited-functionality software it does not have a capability to create a file, or modify the content of an existing one. Instead, it is used only to display or print the content. File viewers do not edit files, but they are able to save data in a different file format.

All the fundamental types of file viewers are filters which translate binary files into plain text (one example antiword). Another common type of file viewer is a picture viewer that can display picture files of various formats. Common features here are thumbnail preview and creation, and image zooming.

The primary reason behind limited functionality is marketing and control. For example, a popular software program, Adobe Acrobat, can be used to create content for most computer platforms, under various operating systems. To ensure that people can access the documents created with Adobe Acrobat, the software publisher created a viewer program, the Acrobat Reader, and made it available for free. This viewer application allows the content created by the proprietary authoring software to be readable on all supported operating-system platforms, free of charge, thus making it a more attractive solution.

There are many products which can qualify as a file viewer: Microsoft Word viewer or Microsoft PowerPoint viewer, and the Open Office equivalents are examples. In a sense, a web browser is a type of file viewer, which translates, or renders, the HTML markups into a human-friendly presentation. Although HTML is plain text, viewing an HTML file in a browser and in a text editor produces significantly different results.

Google Docs is another very good example of online file viewer. Google Docs Viewer supports 12 new file types in, including all remaining Microsoft Office file types, Apple's Pages format, and Adobe's Photoshop and Illustrator files.

Computer performance enhancement utilities

A number of utilities are available to improve the overall performance of the computer system by letting you speed up your system or increase storage space. These utilities range from those that come packaged with the operating system or can be purchased separately.

Disk defragmenter utility reorganizes non contiguous files into contiguous files and optimizes their placement on the hard drive for increased reliability and performance.

There are many hardware and software **accelerators** available to enhance performance in a particular area. For example, download accelerators are software tools to increase the download speed, while graphic accelerators are coprocessors that assist in drawing graphics.

The Windows registry can quickly become crowded and hence slower to search when you remove unused programs that do not uninstall properly. There are utilities like Registry Mechanic or Registry Clean Expert that can help **clean Windows registry** to improve performance.

1.6 PERVERSE SOFTWARE

Perverse software is a program which causes hindrances in other programs execution in such a way resulting in modification or complete destruction of data without the user's intention or even sabotaging the operational system.

Perverse Software is also known as Malicious software or malware. It is a type of software that is designed to secretly access a computer system, without the owner’s consent, and damage the system. The impact can be as damaging as shutting down a business, pulling down computer network or significantly impacting regular use of individual computer systems etc. The damage done can vary from something as little as changing the author's name in a document to full control of one’s machine without the ability to easily find out.

Most malware requires the user to initiate its operation. For example, sending infectious attachments (it acts when users downloads them and runs the attachment) in e-mails, browsing a malicious website that installs software after the user clicks ok on a pop-up, and from vulnerabilities in the operating system.

Early infectious programs, such as Internet Worm and MS DOS viruses, were written as experiments and were largely harmless or at most annoying. With the spread of broadband Internet access, malicious software has been designed for a profit, for forced advertising. Here the malware keeps track of user’s web browsing, and pushes related advertisements.

Typical types of malicious software are - Computer virus, Computer Worm, Trojan horse, Rootkits, Spyware etc.

Here is brief information about the various types of malware:

Computer Virus

Computer virus is a small software program that is designed to enter a computer without users' permission or knowledge, to interfere with computer operation and to spread from one computer to another. A computer virus needs to attach itself to a document or program to infect other computers or programs.

Some viruses do little but replicate while others can cause severe harm or adversely effect program and performance of the system. They can destroy files, software, program applications, and cause the loss of data.

There are various types of computer virus that can be classified by their origins, techniques of attack, modes of spreading, forms of infections, hiding locations and the kind of damage caused. Examples of computer viruses are: Randex, Melissa.A and Trj.Reboot

Computer Worm

Computer Worm is a program that is very similar to a virus. It has ability to self replicate. It actively spreads itself over the network, copies itself from one disk drive to another or copies using email. It does not need user action to start it unlike virus. Examples of worms include: PSWBugbear.B, Lovgate.F, Trile.C, Sobig.D, Mapson.

Trojan Horse

When a program is disguised as something interesting and desirable, users are tempted to download and install it on their machine, without knowing what it does. This is when it does the damages by deleting files from the system or by further installing unwanted software. This is the typical technique of Trojan horse. For example, a file called "saxophone.wav file" on the computer of user who's interested in collecting sound samples may actually be a Trojan Horse. Trojan Horses unlike viruses do not reproduce by infecting other files, nor do they self-replicate like worms, but they are extremely dangerous to users computer's security and personal privacy. They make a computer susceptible to malicious intruders by allowing them to access and read files.

Rootkits

This is a technique using which the malware remains concealed in the system, and continues to do the damage in a concealed manner. Rootkits can prevent a malicious process from being visible (ex Task Bar in Windows operating system)

in the list of running applications. Rootkits normally attempt to allow someone to gain control of a computer system. These programs are usually installed by trojans and are generally disguised as operating system files.

Trap doors

This is a way of bypassing normal authentication procedure (windows/ operating system user name and password) to access a system. Once a system is compromised (impacted by) by malware, one or more backdoors may be installed for easier future access to the system.

Logic Bombs/ Time Bombs

Logic Bombs are not programs in their own right but rather camouflaged segments of other programs. They are not considered viruses because they do not replicate. But their objective is to destroy data on the computer once certain conditions have been met. Logic bombs go undetected until launched, and the results can be destructive. For example, some malicious programs are sot off during days such as April Fools Day or Friday the 13th.

Spyware

While so far we have discussed the malware's intent to damage the computer system, spyware is designed for commercial gain. These programs gather information about the user in a concealed manner, show pop-up advertisements, redirects the search engine results to paid advertisements etc.

Keystroke loggers

This is a program, once installed on the system, which intercepts the keys when entering the password or the Credit Card number while shopping online. This can be used for Credit Card fraud.

Data-stealing

This is a web threat that results in stealing of personal and proprietary information to be used for commercial gains either directly or via underground distribution. Some popular examples of recent data-stealing cases are – steal and sell large number of credit card numbers from businesses such as TJX, OfficeMax, Sports Authority etc.

1.6.1 Ways to Counter Perverse Software

Some common ways of countering Malware are as following:

- Ensure that the operating system and any program one uses are up to date with patches/updates.

- Block unwanted email viruses by installing a spam filter and spam blocker.
- When browsing the internet, always watch what one clicks and installs. Do not simply click OK to dismiss pop-up windows.
- Install anti-virus software; scan and update regularly. It can, in most cases, remove and prevent viruses, worms, trojans, and (depending on the software) some spyware.
- Install anti-spyware/anti-adware; scan and update regularly. It will remove and (depending on the software) prevent future adware and spyware.

1.7 OPEN SOURCE SOFTWARE

Open Source Software (OSS) is software that comes with source code, and importantly also provides rights (typically reserved for copyright holders) to study, change and improve the software. This development happens in a larger collaborative environment, without any direct objective of the software's commercial success.

Primary objectives of the Open Source movement are as following:

- Encourage innovation at the grass-root level and facilitate collaborative software development involving individual talent than it being the prerogative of the large companies.
- Reduce the software cost.
- Improve quality and security
- Avoid forced lock-in to vendor's proprietary software.

Open Source Initiative (OSI) is the patron of the Open Source Definiton (OSD) and is the community-recognized body to evaluate and approve the software as OSD compliant. Some key criterion for OSD compliance are mentioned below:

- **Free Redistribution :** The license should allow any party to sell or give away the software as a component of a larger software distribution containing programs from multiple sources. The license shall not require a royalty or other fee for such sale.
- **Source Code :** The program must include source code, and must allow distribution in source code as well as in executable form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.
- **Derived Works :** The license must allow changes to the existing source code and must allow them to be distributed under the same terms as the license of the original software.
- **No Discrimination against specific applications :** The license must not restrict anyone from making use of the program in a specific scenario. For

example, it may not restrict the program from being used in a business, or from being used in drug research.

- **License must Not Be Specific to a Product :** The rights attached to the program must not depend on the program being part of a particular software distribution.
- **License must Not Restrict Other Software :** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Some examples of Open Source Software are:

Programming language

- PHP - Scripting language suited for the web

Operating System

- GNU Project — "a sufficient body of free software"
- Linux — operating system kernel based on Unix

Server Software

- Apache — HTTP web server
- Tomcat web server — web container
- MySQL – database, popular for applications built on LAMP stack (Linux, Apache, MySQL, PHP/PERL/Python)
- MediaWiki — wiki server software, the software that runs Wikipedia

Client software

- Mozilla Firefox — web browser
- Mozilla Thunderbird — e-mail client

Some typical challenges that used to be associated with the Open Source Software were lack of product support that typically comes with proprietary software, future upgrades, end-user training etc. Over a period of time, industry has evolved to overcome these challenges. For example, Red Hat Linux sells Linux operating system and provides product support, training as well. Further, it is important to note that Open Source Software is not always the best option for all the business needs. However, it does provide a good alternative to the proprietary software. One needs to do the required due-diligence to decide the right product for a specific situation.

Check Your Progress 3

1. Differentiate between open source and proprietary software?

   ……

   ……

   …….

2. Identify open source software from the following list?

   a. OpenOffice

   b. Filezilla

   c. MS Word

   d. Pidgin

   e. Confluence

   ……

   ……

   …….

3. What measure should you take to safeguard your computer from a virus attack?

   ……

   ……

   …….

4. Name a few computer performance enhancement utilities?

   ……

   ……

   …….

1.8 SUMMARY

Software is the brain of computer systems. Any piece of hardware is useful till it has its associated software.

In this unit, we studied how software evolved over the years. Software grew in its size, usage, complexity, development techniques, design and architecture. In the early days software was developed for large centralized systems. It was generally bundled with the hardware, since the cost of software was negligible in comparison to the hardware. The source code was easily available for modification, improvement and redistribution. As the hardware developed, machines became smaller yet much more powerful, computer systems usage and software complexity increased considerably. Higher complexity resulted in higher software costs. Software now came at a price and with restrictive licenses.

One could no longer use, modify or redistribute it freely. Various licensing models evolved based on whether the software would be used by an individual or limited users or enterprise wide, whether it is for commercial or non commercial purposes, whether it is free software or proprietary software, or how often it is used. Companies have been selling these licenses to increase their revenues. The changing pattern of software distribution and sharing led to a movement for free and open source software.

We also learnt about different types of system, programming, application and utility software. We learnt how each one can be useful either to enhance a computer systems performance or to improve our productivity and efficiency in all our jobs. Availability of such wide variety of software makes computer systems infinitely useful in all kinds of work areas.

1.9 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 1

1. a) In the Mainframe architecture all operations and functionality are contained within the central (or "host") computer. Users interact with the host through 'dumb' terminals which transmit instructions, by capturing keystrokes to the host and display the results of those instructions for the user.

   File Sharing architecture is network (LAN) based where ‘intelligent’ PC’s or workstation’s downloads files from a dedicated "file server" and then runs the application (including data) locally.

   b) In Client Server architecture, the Client software requests for the service and Server software provides the service. The client and the server software may be on the same machine or two different networked machine.

   In the distributed systems, different parts or components of an application run on different networked machines. There are set of standards that specify how different distributed components communicate.

   c) In Structured programming statements are organized in a specific manner to minimize error or misinterpretation. It enforces logical structure of the program. Here large routines can be broken down into smaller, modular routines. It discourages GOTO statements.
   Non Structured programming is the earliest programming paradigm in which program usually consists of sequentially ordered commands, or statements, usually one in each line. It does not enforce any logical

structure of the program. Its needs discipline on programmers part to write readable and understandable code. Here the whole code is written in one module. It makes extensive use of GOTO statements that leads to spaghetti code.

2. a) **Software Reusability:** Ability of a computer program to be used repeatedly with little or no modifications in many different applications. For example code to authenticate credit card information can be used at all the places where payment is through credit card.

b) **Software Reliability:** Ability of a computer program to perform its intended functions and operations for the specified period of time, in the specified system's environment, without experiencing any failure. The less there is breakdown of the system, the more reliable it is.

c) **Encapsulation:** Ability to hide data and methods from outside the world and only expose data and methods that are required. It helps in hiding all the internal details from outside world. It also provides a way to protect data from accidental corruption

3. Software-as-a-Service (SaaS) is basically a software delivery model where customers can use the software application as a service on demand and pay for it per usage. It is based on the concept of renting application functionality from a service provider rather than buying, installing and running the software yourself.

Cloud computing is the broader concept of using the internet to allow people to access the technology enabled services. Those services must be 'massively scalable' to qualify as true 'cloud computing'.

Cloud computing is basically what SaaS applications run on.

4. With advances in networking technology, vendors began to introduce non-perpetual licensing models, such as subscription or pay-per-use licensing. In the pay-per-use, user is charged each time he/she uses the software, service or module and user does not own the software, rather uses it at on rent for the limited period. There is time based pay-per-use arrangement and transaction based pay-per-use arrangement.

In a time based pay-per-use arrangement, consumers are charged for the amount of time that they used non-owned copies of the software.

In a transaction based pay-per-use arrangement, usage charges occur because a software module has been used. The duration of use is irrelevant.

Check Your Progress 2

1. a) A system software is any computer software which manages and controls computer hardware so that application software can perform a task. Operating systems, such as Microsoft Windows, Mac OS X or Linux are prominent examples of system software. System software is an essential part of computer operations. Application software is a program that enable the end-user to perform specific, productive tasks, such as word processing or image manipulation.

   b) Compiler is a program the converts a source code in high level language to the object code in low level language.

   Linker is a program that uses multiple object files created by the compiler and predefined library object files, links them together and creates a single executable file.

   c) Compiler is a program that takes the whole source code in high level language and converts it into the source code in low level language. Any errors are reported at compile time for the complete code. Once the translation is complete, only the executable version of the code runs in the memory.

   An interpreter takes the source code in high level language one line at a time during run time, translates the instruction into low language code and executes it before proceeding to the next instruction. Hence the interpreted program remains in the source language and is converted into low level language only at run time. So the translator program also needs to be in the memory at run time.

   Since the compiler translated the whole program before it is run while interpreter translates one line at a time while the program is being run, compiled programs run faster than the interpreted ones.

2. Examples are as follows:
   a. Decision Making Software - Expert Choice, Decision Manager
   b. Education Software – Jumpstart, Reader Rabbit
   c. Industrial Automation Software – Computer aided manufacturing (CAM), Programmable Logic Controller
   d. Mathematical Software - Mathcad, Matlab
   e. Simulation Software – OpenModelica, Circuitlogix

3. The device driver for the printer may not have been installed. You can search for the driver for the particular printer on the internet and install it on your machine.

4. a. Text Editor (for ex TextPad)
   b. Device Driver for the scanner
   c. Operating System (for ex Windows Vista)
   d. Database Software (for ex MS Access)

Check Your Progress 3

1. Proprietary software refers to any computer software that has restrictions on any combination of the usage, modification, copying or distributing modified versions of the software. It is owned by an individual or a company (usually the one that developed it). Its source code is almost always kept secret. Advantages of proprietary software include: 1) Availability of reliable, professional support and training; 2) Packaged, comprehensive, modular formats; and 3) Regularly and easily updated. The disadvantages are: 1) Costly, and 2) has closed standards that hinder further development.

   Open source refers to a program in which the source code is available to the general public for use and/or modification from its original design free of charge. Open source sprouted in the technological community as a response to proprietary software owned by corporations. Advantages of Open source are: 1) Low cost and no license fees; 2) Open standards that facilitate integration with other systems; and 3) it is easily customizable. The disadvantages are: 1) Lack of professional support; 2) Evolving developer communities; 3) Lack of release co-ordination; and 4) Erratic updates.

2. OpenOffice (Word Processing Software), Filezilla (FTP Software), Pidgin (Instant Messaging Software)

3. a Install anti virus and anti spyware. Scan and update regularly.
   b. Keep the windows system updated with patches and updates.
   c. Browse and click only known and secure web sites. Avoid suspicious ones.
   d. Open email attachments from verified source only.

4. TweakVista, Boost Windows 2009, Registry Cleaner, WinUtilitites, System Optimize Expert.

1.10 FURTHER READINGS

Information Technology The Breaking Wave By Dennis P. Curtin, Kim Foley, Kunal Sen, Cathleen Morin, Tata- McGraw-Hill Edition.

Introduction to Computers By Peter Norton, Sixth Edition Tata McGraw-Hill Publishing Company Limited.

Computer for Beginners By Er. V. K. Jain, Pustak Mahal.

Introduction to Computer Science By I. T. L. Education Solutions Limited.

http://www.opensource.org/.

http://en.wikipedia.org/wiki/Malicious_software.