Contents

[Module 1 5](#_gjdgxs)

[**Computer Basics** 5](#_30j0zll)

[**Memory and Storage Units** 6](#_1fob9te)

[**The Arithmetic/Logic Unit** 8](#_3znysh7)

[**The Operating System** 9](#_2et92p0)

[**Programming Languages** 10](#_tyjcwt)

[**Computer Hardware** 10](#_3dy6vkm)

[**Networking** 11](#_1t3h5sf)

[**References** 12](#_4d34og8)

[Module 2 14](#_2s8eyo1)

[**Introduction to Computer Programming** 14](#_17dp8vu)

[**Programming Languages** 14](#_3rdcrjn)

[**Algorithm** 14](#_26in1rg)

[**Flow Chart** 15](#_lnxbz9)

[**Elements of Programming Languages** 16](#_35nkun2)

[**Getting Started!** 17](#_1ksv4uv)

[**Data Types & Type Casting** 18](#_44sinio)

[**Type Casting** 19](#_2jxsxqh)

[**Operators** 19](#_z337ya)

[**Arrays & Strings** 20](#_3j2qqm3)

[**Files I/O in C Programming:** 21](#_1y810tw)

[**Object Oriented Programming** 23](#_4i7ojhp)

[Module 3 26](#_2xcytpi)

[**SDLC (Software Development Life Cycle)** 26](#_1ci93xb)

[**Objectives of SDLC** 26](#_3whwml4)

[**SDLC Models** 31](#_2bn6wsx)

[**Agile Model** 35](#_qsh70q)

[**Sprint** 36](#_3as4poj)

[Module 4 39](#_1pxezwc)

[**Introduction to Web Programming** 39](#_49x2ik5)

[*How did Web Programming begin?* 39](#_2p2csry)

[**The Internet** 39](#_147n2zr)

[**Browsers** 39](#_3o7alnk)

[**Social Networking Sites** 40](#_23ckvvd)

[**Concepts of HTML** 42](#_ihv636)

[**How is HTML written?** 44](#_32hioqz)

[**HTML – Formatting** 44](#_1hmsyys)

[**JavaScript** 48](#_41mghml)

[Module 5 53](#_2grqrue)

[**XML – An Overview** 53](#_vx1227)

[**Overview of XML** 53](#_3fwokq0)

[**Main Characteristics of XML** 53](#_1v1yuxt)

[**Getting Started with XML** 54](#_4f1mdlm)

[**Tags and Elements of XML** 55](#_2u6wntf)

[**Points to remember:**](about:blank) 55

[**XML Documents** 56](#_19c6y18)

[**XML Declaration** 60](#_3tbugp1)

[**Document Type Declaration** 60](#_28h4qwu)

[**XML Tree Structure** 61](#_nmf14n)

[**XML Schemas** 64](#_37m2jsg)

[Module 6 67](#_1mrcu09)

[**Core Concepts of Relative Database Management Systems (RDBMS)** 67](#_46r0co2)

[**File Management System** 67](#_2lwamvv)

[**Database Management System (DBMS)** 67](#_111kx3o)

[**Types of Database Models** 68](#_3l18frh)

[**What is RDBMS?** 69](#_206ipza)

[**RDBMS Characteristics** 69](#_4k668n3)

[**Database Designing** 69](#_2zbgiuw)

[**E-R (Entity Relationship) Modeling** 70](#_1egqt2p)

[**Normalization** 71](#_3ygebqi)

[Module 7 77](#_2dlolyb)

[**SQL Basics** 77](#_sqyw64)

[**Concepts of SQL:** 77](#_3cqmetx)

[**Where is SQL used?** 77](#_1rvwp1q)

[**SQL Language Elements** 78](#_4bvk7pj)

[**SQL Operators** 78](#_2r0uhxc)

[**Classes of Operators** 79](#_1664s55)

[**Type of Operators** 79](#_3hv69ve)

[**SQL Statements or Commands** 81](#_1x0gk37)

[**DQL - Data Query Language:** 83](#_4h042r0)

[**SQL Integrity Constraints** 87](#_2w5ecyt)

[**PRIMARY Key:** 88](#_1baon6m)

[**FOREIGN Key:** 89](#_3vac5uf)

[SQL Other Topics 90](#_2afmg28)

[**SQL Joins** 90](#_pkwqa1)

[**SQL Join Types***:* 90](#_39kk8xu)

[**SQL Indexes** 92](#_1opuj5n)

[**SQL - Using Views** 93](#_48pi1tg)

[**SQL Transactions** 93](#_2nusc19)

[**Transaction Control** 94](#_1302m92)

[**The COMMIT and ROLLBACK Command:** 94](#_3mzq4wv)

[**The SAVEPOINT Command:** 94](#_2250f4o)

[Module 8 96](#_haapch)

[UML Basics 96](#_319y80a)

[**Concepts of UML:** 96](#_1gf8i83)

[**A Conceptual Model of UML** 97](#_40ew0vw)

[**UML Diagrams** 97](#_2fk6b3p)

[**Structural UML Diagrams:** 98](#_upglbi)

[**Package Diagram** 98](#_3ep43zb)

[**Object Diagram** 99](#_1tuee74)

[**Composite Structure Diagram** 101](#_4du1wux)

[**UML Behavioral Diagrams:** 102](#_2szc72q)

[**Sequence Diagrams:** 103](#_184mhaj)

[**Use Case Diagram:** 104](#_3s49zyc)

[**Communication Diagram** 105](#_279ka65)

[**Interaction Overview Diagram** 106](#_meukdy)

[**Interaction Overview Diagram for Sale Process:** 107](#_36ei31r)

[Module 9 109](#_1ljsd9k)

[**UNIX Fundamentals** 109](#_45jfvxd)

[**The UNIX Operating System** 109](#_2koq656)

[**UNIX Architecture** 109](#_zu0gcz)

[**Kernel** 109](#_3jtnz0s)

[**Files and Directories** 110](#_1yyy98l)

[**Home Directory** 113](#_4iylrwe)

[**Directory Structure** 114](#_2y3w247)

[**UNIX File Permission Setup** 115](#_1d96cc0)

[**UNIX Environment** 117](#_3x8tuzt)

[**UNIX Communication** 117](#_2ce457m)

[**Operation Modes:** 119](#_rjefff)

[**User Administration** 119](#_3bj1y38)

[**Managing Users and Groups:** 120](#_1qoc8b1)

[Module 10 121](#_4anzqyu)

[Software Testing 121](#_2pta16n)

[**Quality Assurance** 121](#_14ykbeg)

[**Testing Levels** 126](#_3oy7u29)

[**Non-Functional Testing** 128](#_243i4a2)

[**Testing Methods** 129](#_j8sehv)

[**White-Box Testing** 130](#_338fx5o)

[**Grey-Box Testing** 131](#_1idq7dh)

[**Version Control Systems** 131](#_42ddq1a)

[Module 11 136](#_2hio093)

[**Cloud Computing** 136](#_wnyagw)

[**Benefits of Cloud Computing** 136](#_3gnlt4p)

[**Cloud Computing Service Models** 137](#_1vsw3ci)

[**Cloud Computing Deployment Models** 138](#_4fsjm0b)

[**Cloud Computing-Architecture** 140](#_2uxtw84)

[**Infrastructure as-a-Service (IaaS)** 140](#_1a346fx)

[**Software-as-a-Service (SaaS)** 142](#_2981zbj)

*Copyright © 2019 by Vijayamohan Rao Vatti*

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed “Attention: Permissions Coordinator,” at the address below.

***About the Authors:***

***Vijay Rao:*** *Vijay possesses nearly two decades of experience in providing strategic guidance and identifying/evaluating business strategy across all industry verticals. He is best described as “a multi-faceted personality and ‘class A+ act’ who strives for success.” Mr. Rao has a strong educational background with a Masters (MIS) from Boise State University and is an avid reader in English, Hindi, and Telugu.*

*Vijay was born in India and currently resides in Chandler, Arizona along with his wife and their 2-year-old daughter.*

***Mansi Shanbhag:*** *Mansi comes with extensive collective experience at creating and deploying innovative training solutions covering 500+ programs all across India over the past 15 years. She provides a cutting-edge approach to corporate and organizational training, both outsourced and on-request for consulting services.*

*Mansi currently resides in Pune, India and enjoys listening to music and spending time with her kids, Mohit and Mihika.*

# Module 1

## **Computer Basics**

***Let us first begin with some basic Computer Knowledge***

The word “computer” has been derived from the Latin term “*Computare*” which means to calculate. However, as you may know, the computer can do much more than just simple calculations.

In basic terms, we can define a computer as an electronic device or machine which acts as an information processor: it takes in raw information (or data) at one end, stores it, and then operates on it or processes it to provide the required result at the other end.

The retrieval of information is known is better referred to as an **input**. Storing information is known as **memory** (or storage), and operating on the information is also known as **processing**. The end result of these function becomes none other than the **output**.

**The Functional Units of a Computer**

The computer functions can be divided into the following 5 components:

* Take Input Data
* Store Data
* Process Data
* Output the Information
* Control the Workflow

To provide the above functionality, the computer has the following 4 units:

***Input Devices***

The computer can receive information from users from input devices. A user can enter information using a number of input devices like a keyboard, mouse or any other device such as a webcam, punch cards, magnetic tapes, joystick, pen-drive, magnetic disk etc.

The input unit accepts information or data from an input device, which is then converted to a machine-readable form, i.e. digital form. This data in the form of digitized information then moves to the Central Processing Unit (CPU).

***Storage***

Storage is the component of a computer that stores data on a long-term basis. Usually, storage in a computer comes in the form of a solid-state drive or a hard drive; this drive is known as the primary storage of the computer. Storage allows the user to access and store applications, operating system information, and files for an indefinite period of time.

Most of the modern personal computers also make use of off-line storage. These media forms are known as secondary and tertiary storage devices. Optical discs and flash memory devices are the most popular off-line storage devices used these days. To a lesser extent, removable hard disk drives are also being used. Magnetic tape is another form of a storage device, although predominantly used in enterprise ventures.

***Output Unit***

The output unit displays the results obtained through the operations of a computer’s central processing unit (CPU). The result may be in hard or soft form i.e. visual display, printout, audio clip etc.

***The Central Processing Unit (CPU)***

All the computing operations are performed at the Central Processing Unit (CPU) of the computer or the “brain” of the computer. It executes the instructions, controls the storage of data, and also coordinates with input or output devices attached to the computer.

The CPU is the most important part of the computer which coordinates with all other components of the computer. The CPU is made up of a very complex and extensive set of electronic circuitry.

The CPU has the following functions to perform:

* CPU is considered the brain of the computer so its main function lies in controlling various functions and parts of the computer.
* CPU performs all types of data processing operations.
* It stores data, intermediates the results, and provides instructions (known as a program).
* A CPU has the following three components:
  + Control Unit
  + Memory or Storage Unit
  + ALU (Arithmetic Logic Unit)

***Control Unit***

The control unit of the CPU contains extensive circuitry, and through generating electrical signals, it directs the entire computer system to carry out, or execute, the program instructions stored inside the memory unit. Like an orchestra leader, it passes the instructions to different parts without carrying out any instruction itself. To understand how the CPU unit works, one has to be very clear about the relationship it has with the data storage or memory.

### **Memory and Storage Units**

The computer has three types of memory units:

* Cache Memory
* Primary Storage
* Secondary Storage

It is very common for new computer users to be confused by which parts of the computer constitute the memory, and also to consider memory and storage to be the same thing. The term memory refers to the component within your computer that allows you to access data that is stored for a short term.

Both the hard drive and RAM are memory; however, it is more appropriate to refer to RAM as primary memory and hard drive simply as storage.

“How much memory does my computer have?” is a basic question this is quite commonly asked. The memory of a computer is often between 1 GB and 16 GB of Random-Access Memory (RAM) and ranges from several hundred gigabytes up to a terabyte of hard disk drive storage. In other words, you always have more hard drive space than RAM. The hard disc or, the secondary memory of the computer, has a large number of data stored on it, hence there is a need for a greater capacity.

Characteristics of Secondary Memories:

* Magnetic and optical memories which are also known as backup memory
* Non-volatile memory
* Data is permanently stored even if power is switched off
* Stores large data in a computer
* Computer may run without secondary memory
* Access to memory is slower than primary memories

***ROM (Read Only Memory)***

Memory with text we can only read (we cannot write on it)

* Non-volatile memory
* Information permanently stored in memories during manufacture
* A ROM, stores instructions that are required to start a computer. This operation is referred to as bootstrap

***RAM (Primary Memory)***

For computer’s functioning, however, the CPU needs to interact very closely with the primary storage, or main memory, both for instructions and retrieving the data. A computer can work without secondary memory but not without primary memory. Though the primary memory is not exactly part of CPU, it can be called an extension.

This part of a computer’s memory is used for a small amount of time, only when the program is running. The main characteristic of this type of memory is to allow quick access to the required instructions and data. Since its individual memory cells can be accessed in any sequence, it is also known as the random-access memory (RAM), which is the primary memory of a computer. This RAM holds the data and instructions only of that program on which the computer or the CPU is currently working.

* RAM (Random Access Memory) is the internal memory of the CPU for storing data, program and program results.
* It is a read/write memory which stores data until the machine is working.
* When the machine is switched off, data is erased, so it is a volatile memory.

RAM has two types:

* Static RAM (SRAM)
* Dynamic RAM (DRAM)

SRAM uses transistors to store a single bit of data which is not needed to be refreshed. DRAM uses a separate capacitor for each bit of data, and needs to be periodically refreshed to maintain the charge in the capacitors.

|  |  |
| --- | --- |
| SRAM | DRAM |
| It has long life, no need to refresh  Faster,  Used as cache memory  Large size  Expensive, high power consumption | It has short data lifetime  Needs to be refreshed continuously  Used as RAM  Less in Size  Less expensive, less power consumption |

In modern computers, RAM is organized into modules that can be upgraded easily. This would allow increasing the RAM capacity or fixing damages very easily.

***Cache Memory***

The CPU may also have access to cache memory which is like a very fast (and expensive) version of RAM that allows the CPU to have information that it needs close at its hand quite frequently.

* Cache memory is a special memory used by the CPU for the purpose of decreasing the average time taken for memory accesses. Cache memory is a relatively smaller sized memory capability which has been designed to store the most frequently accessed data of the main memory.
* Cache memory is checked first to see whether the data exists there, so the main memory is not accessed. Typically, there are separate caches for data and instructions. Data cache is typically set up in a hierarchy of cache levels (sometimes called multi-level caches). L1 (Level 1) and L2 (Level 2) are the top most caches in this hierarchy of caches. L1 is the closest cache to the main memory and is the cache that is checked first. L2 cache is the next in line and is the second closest to main memory. L1 and L2 vary in access speeds, location, size, and cost.

### **The Arithmetic/Logic Unit**

An arithmetic logic unit (ALU) is a digital circuit which is used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit of a computer. Most modern CPUs consist of mentioned earlier.

Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. It is the control unit which tells the ALU what operation is to be performed on which data and when it is to be performed. When the processing of the data is over, the ALU stores the result in an output register. The function of the control unit is to move the data between the registers, the ALU, and the memory.

***How does an ALU Work?***

An ALU performs basic arithmetic and logic operations. Such features include addition, subtraction, multiplication, and division. Logic operations are those which are involved in comparisons of values using NOT, AND, and OR.

* The arithmetic/logic unit (ALU) contains the electronic circuitry that executes all arithmetic and logical operations.
* The control unit fetches (gets) the instruction from memory.
* The control unit then decodes the instruction (decides what it means) and directs the necessary data to be moved from memory to the arithmetic/logic unit.
* The arithmetic/logic unit executes the arithmetic or logical instruction. The CPU gives the commands, whereas the ALU performs the actual operation on the data.
* The arithmetic/logic unit stores the result of this operation in memory or in an output register.
* The control unit eventually directs memory to release the result to an output device or a secondary storage device.

Just as there are many different languages that people understand, each different type of CPU has its own instruction set that it understands. Therefore, one CPU of a Compaq personal computer cannot understand the instruction set from another CPU, say for instance, a Macintosh.

### **The Operating System**

An operating System (OS) of a computer can be defined as an interface between a computer user and the computer. It is an integrated set of specialized programs that are used to manage overall resources and operations of the computer.

It is a specialized software that controls and monitors the execution of all other programs residing in the computer. It provides a software base or platform on which the different application programs and the system software run.

An operating system has been designed to perform and manage all the tasks of a computer like file management, memory management, process management, input and output management, along with controlling all peripheral devices such as disk drives, printers and scanners.

Linux, Windows, OS X etc. are the names of some popular operating systems.

***Functions of an Operating System***

An operating system has the following functions to perform:

* Process Management
* Memory Management
* Device Management
* File Management
* Security
* System Performance
* Job Accounting
* Error Detection and Troubleshooting
* Coordination between Software and Users

***Classification of Operating Systems***

Depending upon the needs of the computer, there can be various types of operating systems:

**Multi User**: This OS allows two or more users to run programs at the same time. Some operating systems permit hundreds or even thousands of concurrent users.

* **Multiprocessing**: Supports the running of the same program on more than one CPU.
* **Multitasking**: This OS has the capacity to allow more than one program to run concurrently.
* **Multithreading**: This OS allows different parts of a single program to run concurrently.
* **Real time**: The real time OS responds to input instantly.

General-purpose operating systems, such as DOS and UNIX, are not real-time A real-time operating system does not necessarily have to be fast; it simply has to be quick enough to respond to inputs in a predictable way. Embedded computers often contain RTOS (real time operating system) which are used for controlling a particular task.

***User Interface***

The user interacts with the operating system through a set of commands. For example, like Copy or Delete, the command processor accepts and executes these commands. Graphic user interfaces (UI) allows the user to enter commands by clicking at objects appearing on the screen.

### **Programming Languages**

A programming language is a set of grammatical rules (syntax) and vocabulary (keywords that a computer understands) for giving instructions to perform specific tasks.

Programming languages can be used to create programs to control the behavior of a machine through accurate algorithms, very much similar to the human communication process. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, Java, FORTRAN, Ada, and Pascal.

Regardless of what programming language one uses, it is essential for that program to be converted into a machine language so that the computer can understand it. There are two ways to do this:

1. Compile the program
2. Interpret the program

***Assembly Languages***

Lying between machine languages and high-level programming languages, there are languages which are called assembly languages. Assembly languages are similar to machine languages, but they are much easier than programming languages because they allow a programmer to substitute names for numbers. Machine languages consist of numbers only.

***4GL (4th Generation Languages)***

Above the high-level languages are languages which are known as fourth-generation languages (usually abbreviated 4GL). 4GLs are far removed from machine languages and represent the class of computer languages that are very close to human languages.

### **Computer Hardware**

Computer hardware is the collection of its parts that can be physically seen and touched. Some hardware components are easy to recognize, such as the computer case, keyboard, and monitor. However, there are many different types of hardware components which are required.

**Ports**

* A port is a physical docking point where an external device can be connected to the computer.
* A port has the following characteristics
  + External devices are connected to a computer using cables and ports.
  + Ports are slots on the motherboard into which a cable of external device is plugged in.

Examples of external devices attached via ports are mouse, keyboard, monitor, microphone, speakers etc. Different types of ports are used for different purposes:

* Serial ports are used for external modems or older mouse.
* Parallel ports are used for scanners and printers.
* USB ports are used for connecting to all kinds of external USB devices such as external hard disk, printer, scanner, mouse, keyboard, etc.
* VGA Port connects monitor to a computer's video card.
* Modem port connects to a telephone network.

### **Networking**

A computer network is a system in which multiple computers are connected to each other to share information and resources. The computer network provides the following benefits:

* Share resources from one computer to another.
* Create files and store them in one computer while providing access to those files through the other computer(s) connected over the network.
* Connect a printer, scanner, or a fax machine to one computer within the network and let other computers of that network also use these resources.
* ***The most common resource shared today is the internet; where one connection can be used by a number of computers.***

***Types of Network Connections***

Different computers (computers, printers, routers or other devices) on a network are known as nodes. The technique or the model in which computers are connected is known as topology. A network topology means the pattern in which nodes are connected to a local area network (LAN) or another network. Nodes are connected via links (e.g., twisted pair copper wire cable or optical fiber cable).

There are four principal topologies used in LANs: star, bus, ring, and mesh. The most common topology used now-a-days is collapsed ring topology. This is due to the success of a network protocol called the Ethernet. This protocol, or network language, supports the Internet, Local Area Networks, and Wide Area Networks. This topology is one of the most convenient ways to achieve ring redundancy in a network that has multiple nodes laid out in a mostly linear fashion.

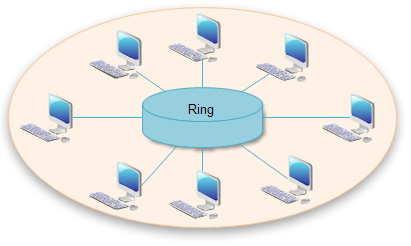
**Star Topology:**

|  |  |
| --- | --- |
| Star Network Topology | This network design has a central node extending cable to each computer on the network so all the computers are connected independently to the network. If a cable is broken, all other computers are able to operate without problems, however this topology requires a lot of cabling. |

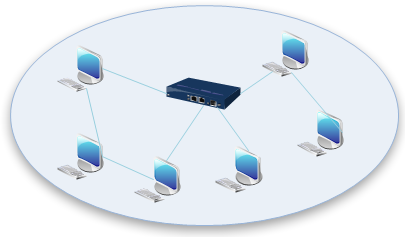
**Bus Topology:**

|  |  |
| --- | --- |
| Bus Network Topology | In this type of design, a single cable connects to all computer nodes, and the information intended for the last node on the network runs through each connected computer. If a cable is broken, all computers connected down the line cannot reach the network. Minimal use of cabling is the benefit of the bus topology. |

**Ring Topology**

In this design, computers are connected via a single cable, but the end nodes are also connected to each other. In this design, the signal circulates through the network until it finds the intended recipient. If a network node is not configured properly, or it is down temporarily for some other reason, the signal will make a number of attempts to find its destination, and other nodes will not be affected. In a collapsed ring topology, the central node in a network is called a hub, a router, or a switch. This network works on a ring topology described above. Each computer has an independent cable, which plugs into the central node or device. Most modern offices have a cabling closet, or a space for keeping the switch device that is connected to the entire network of computers in the office. Even if a network plug is near a desk, the plug is connected via a cable to the cabling closet.

**Mesh Topology**

A network setup where each computer and network device are interconnected with one another, allowing for most transmissions to be distributed, even if one of the connections goes down is known as Mesh technology. A mesh topology can be a full mesh topology or a partially connected mesh topology. It is a topology which is commonly used for wireless networks. Now- a-days it is possible to do networking without cables, through radio waves.

### **References**

* <http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm>
* <https://www.computerhope.com/jargon/m/memory.htm>
* <http://study.com/academy/lesson/arithmetic-logic-unit-alu-definition-design-function.html>
* [www.wu.ece.ufl.edu/links/dataRate/DataMeasurementChart.htm](http://www.wu.ece.ufl.edu/links/dataRate/DataMeasurementChart.htm)
* <https://www.tutorialspoint.com/operating_system/os_overview.htm>
* <http://www.webopedia.com/TERM/O/operating_system.html>
* <http://www.webopedia.com/TERM/P/programming_language.html>
* <https://www.tutorialspoint.com/computer_fundamentals/computer_ports.htm>
* <https://www.quora.com/What-is-the-meaning-of-port-in-computer>
* <http://study.com/academy/lesson/what-is-a-computer-network-types-definition-quiz.html>
* <https://www.computerhope.com/jargon/m/mesh.htm>
* <https://www.edrawsoft.com/Network-Topologies.php>

# Module 2

## **Introduction to Computer Programming**

***How Can Programming Change Your World***

A computer program is a sequence of precise instructions written for a computer to enable it to perform a specified task. All computer programs are written using specific Computer Programming Languages. A computer without a computer program is just a dump box; it is programs that make computers work.

* Computer Programming is the process of developing and implementing various sets of instructions that enable a computer to do a specified task. These instructions, as previously mentioned, are computer programs that help the computer to perform specific computer operations smoothly.
* It is a process that leads an original formulation of a computing problem to an executable computer program.
* Programming involves many activities such as analysis, developing and understanding the specific task, and then generating algorithms (a set of instructions that tell your computer precisely what steps to take to solve a problem). Programming also involves verification of requirements of algorithms along with their correctness, resource consumption, and implementation.

Computer programming is a part of a big discipline known as software engineering, which comprises several different aspects of making software including: design, construction and quality control.

A computer program is also called a computer software, which can range from a mere ‘two lines’ to ‘millions of lines’ of instructions. The instructions in a program are also called program source code and the computer programming is also known as program coding.

MS Word, MS Excel, Adobe Photoshop, Internet Explorer, Chrome, etc.are examples of computer programs.

### **Programming Languages**

A programming language which is used for writing a program for a computer is different from the other languages we know; a programmer has to specifically learn these programming languages for writing a computer program.

Writing different types of programs requires different programming languages which can range from a high-level programming language to writing directly in a low-level machine code. (Code is something which directly controls the essentials of the computer's hardware, a language which is understood by the hardware of a computer).

### **Algorithm**

An algorithm is a list of well-defined instructions for calculating a function in mathematics and computer science. Algorithms can be used for calculation, data processing, and automated reasoning. To say it simply, Algorithm is a finite set of well-defined and precise instructions― a step-by-step procedure to tell the computer how to solve a problem or reach a goal. Algorithm is a recipe of sorts where the ingredients are called inputs, while the results are called the outputs.

An algorithm can be written to do just about anything such as:

* Arrange a list of names alphabetically
* Sort the books per author
* Find the cheapest hotel in a particular city etc.

For any of the above problem-solving exercises, precise instructions or algorithms can be written. Take the example of multiplication, which is adding the same number that many times; but instead of adding we multiply. So, the algorithm can also be defined as the most efficient (not always) way to solve the problem.

E.g. algorithm to find out the largest number from a given list of numbers will be written as:

* Get a list of numbers L1, L2, L3....LN
* Assume L1 is the largest
  + Largest = L1
* Take the next number Li in the list and do the following:
  + If Largest L1 is less than Li
    - Largest = Li
  + If Li is the last number from the list:
    - Print value stored in Largest and come out
  + Else, repeat the same process starting from step 3

### **Flow Chart**

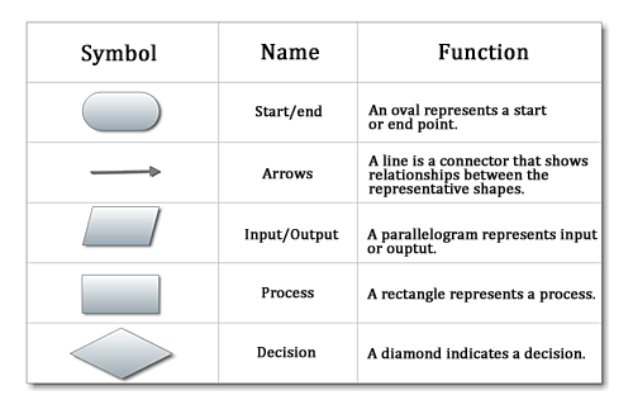
Flowcharts were first developed by Herman Goldstine and John von Neumann in the 1940's. Flowcharts provide a step-by-step diagram for depicting complex situations of a programming code or different troubleshooting problems.

After an Algorithm is written, it is represented by means of a special type of diagram called a Flow Chart. The Flow Chart depicts the workflow, or the process, where different steps are shown using boxes of various kinds connected by arrows representing the direction or the order of the process.

#### **Flow Chart Symbols**

A Flow Chart can also be defined as a graphical depiction of decisions and results represented through individual boxes of varying shapes. These boxes are called Flow Chart Symbols. Flow Chart Symbols represent the various actions of the computer program.

There are many symbols which are frequently used in Flow Charts. The start/end symbol represents either the beginning or the end of a program. The rectangle symbol depicts the process (how the program has to function) like when you need the program to analyze the information or do calculations on numbers.

Input/output symbol represents entering of data, showing it on the screen, or printing it on paper. 

The display symbol signifies that information is to be displayed to the user. The decision symbol is used for things like 'if statements,' where an option is to be chosen based on a specified criterion.

Sometimes, when you make a Flow Chart, a break in the flow of the chart occurs. This is the time when you may have to leave a portion of the chart, and then restart again somewhere else. Like when you reach the bottom of a column, you need to go back to the top of the page to continue with a new column. In such cases the connector symbol is inserted with the same number in it at the exit and re-entry points which helps the reader in understanding the flow and the program logic correctly.

### **Elements of Programming Languages**

#### **Programming Environment**

Programming environment is basically a Software setup, which needs to be installed on a computer to provide a base for writing, compiling, and executing programs. It also includes the development environment which is constituted by the set of processes and programming tools which are needed to create the program or software product.

***Basic Syntax***

Like any language, a computer programming language also makes use of certain rules of grammar and spelling for writing a program known as the Basic Syntax. Syntax means to follow a specific character structures that a computer can understand.

***Data Type***

In computer programming, a data type or simply a type implies a classification of data to tell the compiler or interpreter in the computer how the programmer intends to use the data. Data types like real, integer, or Boolean are used in most programming languages.

***Variables***

In computer programming, a variable denotes a value that can change depending on conditions or the information. Variables are provided to the program during its run.

***Keywords***

Keywords for programming languages are specific words that are reserved by a program because these words have a special meaning. Keywords (also known as reserved names) can also be commands or the parameters, but these cannot be used as variable names.

***Basic Operators***

In computer programming, basic operators are characters that represent actions. For example, + is an arithmetic operator which represents addition and X is an arithmetic operator that represents multiplication. The most familiar sets of operators are the Boolean operators, which are used to work with true/false values.

***Decision Making***

When the programmer specifies one or more conditions to be evaluated or tested by the program, it is called decision-making. Each program has inbuilt statements which are to be executed if the condition is determined to be true, and alternatively other statements to be executed when the condition is determined to be false. Decision-making is done at the junction of the two.

***Loops***

A loop in a computer program denotes a sequence of instructions that repeatedly occur in a program until a certain condition is reached.

***Arrays***

An array can be defined as a collection of variables of the same type. An array is a data structure of a fixed-size that stores a collection of elements of the same data type.

***Strings***

Any finite sequence of characters - letters, numerals, symbols and punctuation marks - is known as a string. Different programming languages use different data types, but string is a data type invariably used by every programming language.

***Functions***

A function is a chunk or piece of organized code that is used to perform a single, related action. Functions can also be defined as reusable codes. Functions provide modularity to different applications.

***File I/O***

When the data is written into a file it is known as file input. Consecutively, when a data is read from a file, it is file output.

### **Getting Started!**

Now you know the terminology or the computer jargon used for programming. Now, let’s get started with a simple but famous 'Hello World' program!

The objective of this code is to print 'Hello World' which tries to show you how C programs are constructed and executed. The program will look like this:

#include <stdio.h>

int main() {

/\* my first program in C \*/

printf ("Hello, World! \n");

return 0;

}

|  |  |
| --- | --- |
| **Basic Commands and Syntax** | **Explanation** |
| #include <stdio.h> | This is a preprocessor command, which tells the C compiler to include stdio.h file before going to actual compilation. |
| int main() | It is the main function where the program execution begins. |
| { | Indicates the beginning of the main function. |
| /\* so me comment\*/ | Whatever is written inside the command /\*...\*/ won’t be considered for compilation and execution. So such lines are called comments in the program. |
| Print ("Hello, World! "); | This command causes the message to be displayed on the screen. |
| The command return 0; | terminates the main() function and returns the value 0 |
| } | This indicates the end of the main function. |

Let us now understand how the above program is written:

1. Open a text editor and add the above-mentioned code. The following steps are to be followed:
   1. Save the file as hello.c
   2. Open a command prompt, then go to the directory where the file was saved.
   3. Type gcc hello.c and press enter to compile your code.
   4. If there are no errors in your code, the command prompt will take you to the next line and would generate a.out executable file.
   5. Now, type a.out to execute your program.
2. You will see the output "Hello World" printed on the screen.

However, to run the above program, the gcc compiler needs to be installed and the program will run in the directory containing the source file hello.c.

### **Data Types & Type Casting**

Data type means the type of data which can be processed through a computer program like numeric, alphanumeric, decimal, etc.

Different programming languages use different keywords to specify different data types:

|  |  |  |
| --- | --- | --- |
| **Key word** | **Data Type** | **Programming Language** |
| Int | Integer (basic) | C and Java |
| Char | Character (basic) | C |
| Float | Fraction | C |
| Typedef | User defined | C |

These data types are called primitive data types and you can use these data types to build more complex data types, which are called user-defined data types.

There are three ways to create user defined data types:

1. Using typedef: Creates new data types.
2. Using Structures: An aggregate data type using integer, float and string.
3. Using Unions: Creates variables to share a common memory space.

### **Type Casting**

Type casting is a way to convert a variable from one data type to another data type. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'. You can convert the values from one type to another.

**Variables & Constants**

* A variable in a computer program is of a particular type (e.g., integer) and it has a name.
* The type of a variable indicates what kind of value it will store.
* The name of a variable is known as its identifier.
* An assignment statement is used to provide a value to the variable. Like float or double is assigned when decimals are needed. Assign char if the variable will always contain ONE letter of data.
* The variable has a particular named memory location that temporarily stores data (whose value can change while the program is running).
* Variables of different types occupy different amounts of memory space and are described based upon the size of the numbers. For example, short, int, or long data types are used when a variable is an integer number (NO decimal points). Only int can be used when it is not short or long.
* Variable can be accessed and updated throughout the entire computer program in which they are declared.
* Scope of a variable is decided by its location in the computer program.
* A constant is a named memory location which temporarily stores data that always remains the same throughout the execution of the program. The value of the constant is never modified.

### **Operators**

An operator is a symbol that tells the compiler to perform certain mathematical, relational or logical operations. Operators are used in computer programs to manipulate data and variables.

* **Arithmetic Operators**: +, -, \*, /, %
* **Relational Operators**: =, =, =!, >, <, >=, <=
* **Logical Operators**: &&, ||

**Operator Meaning of Operator Example**

&& Logical AND True only if all operands are true If c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.

|| Logical OR True only if either one operand is true If c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.

#### **Bitwise Operators**

During computation, in c programming, mathematical operations like addition, subtraction, addition and division, are converted to bit-level operations which makes processing faster and saves power.

<< Shift left >> Shift right

#### **Assignment Operators**

**Operator Example Same as**

= a = b a = b

+= a += b a = a+b

-= a -= b a = a-b

***Conditional Operators***

If this year is a leap year, enter 1. If not, enter any integer: 1

#### **Special Operators**

These operators are used for performing bit operations on any given two variables. Conditional operators return one value if the condition is true or returns another value if the condition is false. These operators are used to either increase or decrease the value of the variable by one.

### **Arrays & Strings**

An array can be defined as a data structure that stores a fixed-size collection of elements or variables of the same data type which share a common name and occupy contiguous memory locations.

The individual elements of an array are referenced by appending a subscript, in square brackets, behind the name. An array represents one of the simplest data structures and are exclusively in embedded systems. Strings are also data structures that are different from arrays in the following ways:

|  |  |
| --- | --- |
| **Arrays** | **Strings** |
| An array can contain fixed number of elements. | The number of elements in a string can vary. |
| Arrays can contain any data type (char, short, int, or even other arrays) | Strings contain ASCII characters which are terminated with a NULL (0) character |
| The individual array elements can be randomly accessed. | The strings are processed sequentially character by character from start to end |
|  |  |

### **Files I/O in C Programming:**

#### **Files and different Types of files**

Computer Files are used for storing data in digital format like plain text, images or other contents. When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates, and with a saved file you can easily move your data from one computer to another without any changes.

Files are organized inside different directories. A directory is a file system or cataloguing structure which contains references to other computer files, and other directories. In some computers, directories are known as folders.

There are two types of files: text files and binary files. A comparison between the two is given below:

|  |  |
| --- | --- |
| **Text Files** | **Binary Files** |
| These are the normal .**txt files** that can be created using Notepad or any text editor. | Binary files in your computer are mostly the **.bin** files. |
| When these files are opened, you'll see the content within the file as plain text which can easily edited or deleted. | Instead of storing data in plain text, they store it in the binary form (0's and 1's). |
| These files take minimum effort to maintain and are easily readable but require greater storage space and provide the least amount of security. | They can hold a higher amount of data, and provides better security than text files, but are not easily readable. |

#### **File Input/Output**

File input means writing data into a file and file output means data that is read from a file (p.19). Both input and output terms are more related to screen input and output. Displaying a result on the screen is called output and entering the data at the command prompt is called input.

Computer files can be considered as the digital counterpart of paper documents. While programming, you keep your source code in text files with different extensions, for example, C programming files end with the extension .c, Java programming files with .java etc.

#### **Working with files**

When working with files, you need to declare a pointer of type file.

***Declaration***

FILE \*fptr; is needed for communication between the file and program.

***Opening a file***

Opening a file is performed using the library function in the "stdio.h" header file: fopen().

The syntax for opening a file in standard I/O is:

ptr = fopen("fileopen","mode")

***Example***:

fopen("E:\\cprogram\\newprogram.txt","w");

fopen("E:\\cprogram\\oldprogram.bin","rb");

Now if the file newprogram.txt doesn't exist in the location E:\cprogram. The first function creates a new file named newprogram.txt and opens it for writing as per the mode 'w'.

The writing mode allows you to create and edit (overwrite) the contents of the file.

Now let's suppose the second binary file oldprogram.bin exists in the location E:\cprogram. The second function opens the existing file for reading in binary mode 'rb'.

The reading mode only allows you to read the file, you cannot write into the file.

**Opening Modes in Standard I/O**

R Open for reading. rb Open for reading in binary mode.

w Open for writing. wb Open for writing in binary mode.

a Open for append ab Open for append in binary mode.

r+ Open for both reading and writing. rb+ Open for both reading and writing in binary mode.

w+ Open for both reading and writing. wb+ Open for both reading and writing in binary mode.

a+Open for both reading and appending. ab+ Open for both reading and appending in binary mode.

#### **Closing a File**

The file (both text and binary) should be closed after reading/writing.

Closing a file is performed using library function fclose().

fclose(fptr); //fptr is the file pointer associated with file to be closed.

***Reading and writing to a text file***

For reading and writing to a text file, we use the functions fprintf() and fscanf(). Both fprint and fscanf expects a pointer to the structure FILE.

Writing to a text file

***Example***: Write to a text file using fprintf()

#include <stdio.h>

int main()

{

int num;

FILE \*fptr;

fptr = fopen("C:\\program.txt","w");

if(fptr == NULL)

{

printf("Error!");

exit(1);

}

printf("Enter num: ");

scanf("%d",&num);

fprintf(fptr,"%d",num);

fclose(fptr);

return 0;

}

This program takes a number from the user and stores in the file program.txt.

### **Object Oriented Programming**

Object Oriented Programming (OOP) is a programming model which is based on the concept of "objects,” rather than actions.

* An object represents the whole class of objects, which is a kind of data structure that includes the data along with the functions.
* Data is organized in fields called attributes.
* The programming is more logic-based than data-based.
* The defined logic sequence, or code, is in the form of a procedure referred to as a method.
* A method also defines the relationship between one object and the other.

The objects inherit characteristics from other objects. This feature makes the OO Programs to be modified easily.

#### **Inheritance**

Inheritance is a feature of OOP which enables the new objects to take on the properties of the existing objects. A superclass forms the basis for inheritance, which is also known as a base class or a parent class. The class that inherits from a super class is called a subclass or a derived class.

#### **Abstraction**

Abstraction is one of three central principles of OOP. It is through the process of abstraction that a programmer can hide all data other than the relevant data about an object. This feature reduces the complexity of the program and increases its efficiency.

#### **Encapsulation**

Encapsulation is the process of binding the data with the code that manipulates it. It is this binding which leads to the concept of data hiding. Encapsulation keeps the data and the code safe from external interference.

#### **Polymorphism**

Polymorphism is the ability of OOPL to process objects based on their class or data type. It is through this feature that the methods get redefined for different derived classes. For example, the programmer can define different area methods for any number of derived or sub-classes, such as squares, circles, rectangles or triangles. Overriding and overloading can also be considered as part of polymorphism.

#### **Overriding**

Overriding enables a sub-class to obtain a different implementation for a method that has already been defined for its parent or super-class. This overridden method should have the same name, signature, and parameters as the one in its parent or super class.

#### **Overloading**

Overloading means creating methods with different parameters but with the same name. Here, the body of a method of super-class is redefined in subclass to change the behavior of the parent method.

#### **Exception Handling**

Responding to exceptions during the running of a computer program is called exception handling. It is a specific means to handle runtime errors so that the program does not get terminated abruptly. The good thing about Java programming is that it is able to handle such exceptions. For example, providing an easily understandable message to the user about the issue in a program, rather than the user seeing a system generated message.

***References:***

1. <https://en.wikibooks.org/wiki/Computer_Programming>
2. <https://en.wikipedia.org/wiki/Computer_programming>
3. <https://www.quora.com/What-is-algorithm-and-how-to-use-it-in-programming>
4. <http://study.com/academy/lesson/flowchart-symbols-in-programming-definition-functions-examples.html>
5. <https://www.adobe.com/devnet/actionscript/learning/oop-concepts/inheritance.html>
6. <https://www.techopedia.com/definition/24010/overriding>
7. <https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html>
8. <https://stackoverflow.com/.../are-polymorphism-overloading-and-overriding-similar-com>
9. <https://fresh2refresh.com/c-programming>
10. <https://www.programiz.com/c-programming/c-operators>
11. <https://www.programiz.com/c-programming/c-file-input-output>
12. <https://fresh2refresh.com/c-programming/c-operators-expressions/c-special-operators>

# Module 3

## **SDLC (Software Development Life Cycle)**

## **Objectives of SDLC**

SDLC (Software Development Life Cycle) is the entire process that is used by the software industry to design, develop, and test high quality softwares. SDLC is also known as ‘Software Development Process’ or ‘Application Development Life-Cycle’.

1. To produce high quality software that meets or exceeds customer expectations; reaching completion within time and cost estimates.
2. To prepare a detailed plan to describe how the specific software will be developed, maintained, replaced, altered, or enhanced in the future.
3. To define the correct methodology for the whole development process along with a strategy to improve the quality of software.

#### **Stages of SDLC**

* Stage 1: Requirement Gathering
* Stage 2: Analysis
* Stage 3: Designing the Product Architecture
* Stage 4: Building or Developing the Product (Coding)
* Stage 5: Testing the Product
* Stage 6: Deploying the Product
* Stage 7: Maintenance (Customer Support)

#### **Stage 1: Requirement Gathering**

Requirement gathering is the most crucial and fundamental stage in SDLC which is to be performed by the senior members of the team.

The inputs are to be gathered from all the stakeholders—the customer, the sales department, market surveys, and domain experts in the industry.

#### **Stage 2: Analysis**

The Analysis phase is a critical phase of SDLC because this is the time when the project leader and the team begin to understand what the customer wants from the project.

***Feasibility Study***

Based on the information gathered, a basic project approach is planned and a feasibility report is to be prepared by the analyst.

Feasibility can be defined as the practical extent to which the given project can be carried out successfully. To find out the feasibility or workability of the project, a feasibility study is performed, that determines whether the solution considered to accomplish the requirements is workable or not.

The following types of feasibility are commonly considered:

***1) Technical Feasibility***

Assessment of the current resources (such as hardware and software) and technology that are required to accomplish user requirements in the project within the allocated time and budget.

***2) Operational Feasibility***

Assessing the extent to which the required software can perform the series of steps needed to solve business problems and user requirements.

***3) Economic Feasibility***

An evaluation on whether the required software is capable of generating enough financial gains for an organization.

The client provides a rough idea about the functionalities and features of the desired software product to be developed. However, it is the job of the analysts to do a detailed study about whether the desired functionality and the system software are feasible to develop or not. The analyst has to study the practicality of the software, its implementation, its financial contribution to the organization, along with its cost constraints. At the same time the analyst has also to explore the technical aspects of the project such as its usability, productivity, maintenance, and its integration ability.

The feasibility study report by the analyst should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

When the project development is accepted, the analyst needs to perform the following:

• Validation of requirement

• Develop a management plan

• Documentation of business

• Documentation of technical, process, and product requirements

• Coordination with clients to remove the ambiguities

• Finalizing acceptance criteria with client and other stakeholders

***Preparation of Software Requirement Specification (SRS) Document***

The next step is to prepare a Software Requirement Specification (SRS) which is mainly intended to be followed by the software development team or to serve as a reference for product architects to develop the best architecture for the product development. This is a highly technical document defining how the intended software will interact with hardware, focusing on the following areas:

• Speed of operation

• Response time of system

• Portability of software across various platforms

• Maintainability of the product

• Speed of recovery after crashing

• Security, Quality, Limitations etc.

***Stage3: Designing***

Based on the specifications in SRS, usually more than one design approach for the product architecture is proposed. A review is done by several stakeholders after considering different parameters such as: risk assessment, product robustness, design modularity, budget and time constraints. Once these items have been reviewed, the best design approach is selected for the product development.

The SDD (software design document) is written by software designers in order to give the software development team an overall guidance or blueprint for the entire architecture of the software project.

***Designing Levels***

The actual software designing is done at three levels:

1) Architectural Design

The architectural design of the software is the most abstract version of the system where the software is described as having many components interacting with each other to build the required product. This level provides the designers the exact idea of the proposed solution domain.

2) High-level Design

At this stage the architectural design is broken into a number of subsystems using the concept of ‘single entity-multiple component’. Less abstraction is achieved at this level by defining the interaction. The aim of High-level design is to focus on how the system, along with its various components, can be implemented in the form of modules.

3) Detailed Design

This is a more detailed version of the design which deals with individual modules and their implementation. Along with defining the logical structure of each module, the design also defines module interfaces, i.e., the way in which a module communicates with other modules. Though the software system is broken into multiple discrete and independent modules, each module is designed to be capable of carrying out task(s) independently.

***Design Verification***

After the final design, the reviewers do a structured verification to detect defects that might be caused by overlooking some conditions. A good design review is very crucial for good software development in terms of accuracy and functionality.

In the final System Design Document, a clear description for the following is given:

* System requirements
* Operating environment
* System and subsystem architecture
* Files and database design
* Input formats and output layouts
* Human-machine interface
* Processing logic
* External interfaces

***Stage: 4 Coding***

It is in this stage of SDLC that the actual development starts. The product is built by generating the programming code as per the SDD blueprint.

Programming code is generated with the help of tools like compilers, interpreters, debuggers etc. Coding is done using different high-level programming languages such as Java, C#, VB.net, C++ and PHP etc. The choice of programming language always depends upon the type of software to be developed.

***Stage5: Testing***

This stage is usually a subset of all the stages as, in the modern SDLC models, testing activities are carried out at almost all the stages of SDLC.

As such, the process of testing runs parallel to the process of software development. Every stage is tested, validated and verified before going to the next stage.

The Software is tested at various levels -

***Unit Testing***

As the name implies, the programmer performs some tests to see whether the individual units of the program are working as per requirement and are error-free.

***Integration Testing***

After unit testing, there is a need to find out if the units, when integrated together, work without error.

***System Testing***

After the different modules of the software are compiled together as a product, then it is tested as a whole. The following tests are performed to see whether the product meets all the specifications.

* ***Functionality Testing***
  + Performed to see if the product meets all the requirements.
* ***Performance Testing*** 
  + This test checks the efficiency of the software. Performance testing is also done under environment conditions where the software is put under high user and data load. The effectiveness of the product is judged by the average time taken by the software to do the desired task.
* ***Security & Portability*** 
  + When the software is supposed to work on various platforms and is to be accessed by a number of persons, the security and portability tests are very essential to be performed.
* ***Alpha Testing*** 
  + Alpha testing implies the testing that is done by a team of developers by simulating the user-work environment, The goal of this testing is to find out how a user would react to some actions in the software and how the system would respond to user inputs.
* ***Acceptance Testing (Beta testing)***
  + After the last phase of testing is done, it is handed over to the customer for testing purposes in their environment. Even when the software matches all user requirements, if the user does not like the way it works, it may be rejected. Developers expect that users at this stage will bring minute problems, which were skipped.
* ***Regression Testing***
  + Whenever a software product is updated with new code or functionality, it is tested thoroughly to detect if there is any negative effect caused by the added code.

***Stage6: Deployment***

* Once the product is tested, it is ready to be deployed or released formally in the appropriate market.
* As per the organization's business strategy, sometimes the product deployment is also done in phases.
* The product may first be released in a limited segment and after UAT- (User acceptance testing), it is released in the real business environment.

***Stage 7: Maintenance***

* After the product is deployed in the market, its maintenance is done for the existing customer base.
* Support may be provided for a fixed number of years as per the contract.
* Maintenance includes problem solving, bug fixing, enhancements to the product, adding new features, etc.

## **SDLC Models**

A software development life cycle (SDLC) model is a blueprint or conceptual framework which describes all activities to be involved in the project throughout the planning and maintenance process. The 7 different stages in development process are streamlined by an SDLC model. There are a series of models which are available; however, the best suitable model is selected based on the specific requirements.

#### **Different SDLC Models**

* Waterfall Model
* Iterative Model
* Spiral Model
* V-Model
* Big Bang Model
* Prototype Model
* Agile Model

***3.1 Waterfall Model***

* This was the first Process Model to be introduced in SDLC.
* It is also known as a linear-sequential life cycle model.
* It is very simple to understand and use.

A characteristic feature of a waterfall model is that in the development process, the next phase can only begin after the current phase is completed so that there is no overlapping.

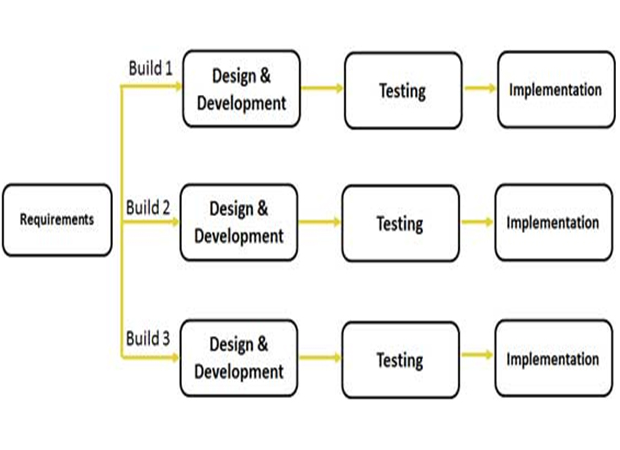
***Advantages of Waterfall Model***

Use of Waterfall model is recommended when:

* The project is short.
* Requirements are well documented, clear, and fixed.
* Product definition is not expected to be changed in the future course of time.
* Technology for the SD is well understood.
* There are no ambiguous requirements in the project.
* Ample resources and required expertise are available as product support.

***Iterative Model***

The basic idea behind this method is to develop a system through repeated cycles (iterations) and in smaller portions at a time (incremental).

The iterative process starts with specifying and implementing a subset of the software requirements. The implemented segment is then reviewed in order to identify further requirements. This process is then repeated to produce a new version of the software for each cycle of the model. 

An Iterative model is a kind of incremental model in which the whole requirement is divided into various builds. During each build or iteration, the development module again goes through the requirements, design, implementation, and testing phases.

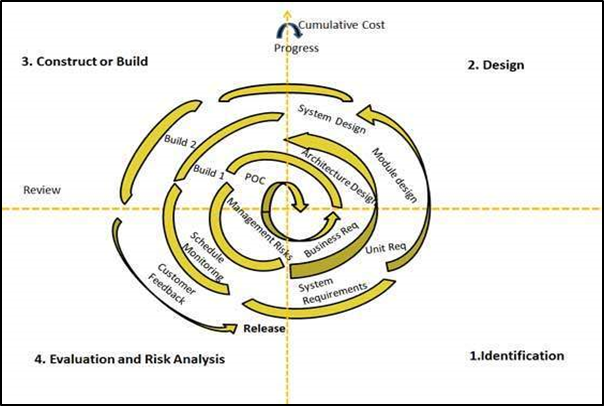
Since the work in this model is done iteratively, the rough product (or the product piece) that is created in earlier iterations is improved in the next iteration. Each iterative cycle enhances the evolving versions until the full system becomes implemented.

#### **Advantages of Iterative Model**

This model is most often used in the following scenarios:

* The requirements of the complete system are clearly defined and understood.
* Though the major requirements are defined, there is scope for some functionalities or requested enhancements to evolve with time.
* There is a time-to-market constraint.
* There may be a requirement for use of a new technology which is to be learnt by the development team while working on the project.
* The resources with needed skill set (though not available), are planned in such a way that they can be hired on a contractual basis during specific iterations.
* There are some high-risk features i.e., possibility of some goals to change in the course of time.

***Spiral Model***

The model is similar to the incremental model, but here more emphasis is placed on risk analysis. The spiral model has four phases discussed below. The software project has to repeatedly pass through these four phases in iterations which are called Spirals. The baseline spiral starts with the planning phase when requirements are gathered and risk is assessed. Each subsequent spiral then builds upon the baseline spiral. Each spiral has the following 4 phases:

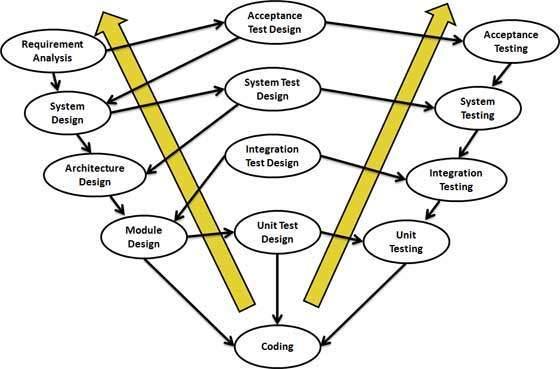
1. Identification
2. Design
3. Construct or Build
4. Evaluation and Risk Analysis

***Advantages of Spiral Model***

The Spiral model is typically used when

* There is a budget constraint and risk evaluation is important.
* The project falls in the medium to high-risk category.
* A long-term project commitment is required because of prospective changes.
* The customer is not too sure of the requirements of the project.
* There are some requirements which are complex and need evaluation to achieve clarity. The product line needs to be released in small phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

***V – Model***



V-Model means Verification and Validation model. This model can also be called an extension of the waterfall model, where each phase must be completed before the next phase begins.

The V-Shaped life cycle symbolizes a sequential path for execution of processes where testing of the product is planned and performed parallel to the development phase.

Since every single phase in the development cycle is directly associated with a testing phase, it is a highly disciplined model.

#### **Advantages of V Model**

V-Model is suitable for use in the following scenarios:

* The requirements are well defined, clearly documented, and are not likely to change.
* The product definition will always remain the same.
* The technology to be used is not dynamic; it is well understood by the project team.
* There is no ambiguity in the requirements.
* It is a short project.

***Big Bang Model***

The Big Bang model is an SDLC model where no specific process is followed for software development. Development is started with the input of required money and efforts and the output of which is software that may or may not fulfil customer requirements.

***Advantages of Big Bang Model***

This model is ideally followed in conditions when:

* The customer is not sure of what exactly they want. Therefore, the requirements are implemented “on the fly” without a great deal of analysis.
* This model is usually followed when dealing with projects where the developmental teams are small in number/resources.

***Prototype Model***

The basic idea to build a prototype model is to understand the client requirements through building a throwaway prototype model based on the current ambiguous requirements. This prototype provides the client with an “actual feel” of the system, enabling them to better understand the requirements of the desired system.

***Advantages of the Prototype Model***

The prototype model can be used when:

* The systems to be developed are very complicated and large for which there is no manual process or existing system available that can be followed.
* When the desired system needs to have a lot of interaction with the end users.
* When user evaluation or feedback for the prototype may be easily available. It is through the user feedback that missing functionality of the product is easily identified.
* There is a requirement in the project for web-based interaction with the users.

## **Agile Model**

Agile methodologies for SDLC came into being after the failure of dominant software development/ project management paradigms. These methodologies evolved when 17 pioneers met at the Snowbird Ski Resort in Utah in 2001 and wrote the Agile Manifesto. In agile methods, more emphasis is given to good communication and collaboration, better organization of team and team members, and providing flexibility in methodologies to adapt to newly emerging business realities.

* The Agile model is a combination of iterative and incremental process models. The model focuses on process adaptability and customer satisfaction by rapid delivery of a working software product.
* The product is broken into small incremental builds, which are used for iterations. The tasks are divided into small time frames to deliver specific features for a release.
* Each iteration typically lasts between one to three weeks (approximately). The agile model provides small incremental releases where each release is built on previous functionality.
* Every iteration cycle involves cross-functional teams who are working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
* At the end of the iteration, a working product is displayed to the customer and other important stakeholders. Each build is incremental in terms of features, and the final build holds all the features required by the customer.

***Advantages of Agile Model***

* Since the agile model believes that every project needs to be handled differently, the existing methods are tailored to suit the project requirements.
* The biggest advantage of an agile model is the high degree of customer satisfaction and the freedom that an agile model provides to accommodate to change. New changes suggested by the customer or users in the system can be implemented at very little cost.
* The Agile model requires very little planning to get started with the project, unlike the waterfall model where extensive planning is required.

This model is ideally used when:

* Close, daily cooperation between business executives and developers is available.
* Regular adaptation to changing circumstances is required.
* Frequent changes in the requirements are anticipated.

There are a number of agile methodologies that are available today; however, the most popular among are Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995).

***Scrum***

Scrum’s philosophy was inspired by the ‘inspect and adapt feedback loops’ to cope with risk and complexity, where decision-making is based on real-world results rather than speculation.

Scrum is an iterative & incremental agile software development method which provides a framework for managing a process through different iterations which are time boxed events.

**Sprint**

Every iteration cycle in the process is called a Sprint. Sprint is a time-box of two weeks or one month, during which a potentially releasable product increment is created. At the end of each sprint, stakeholders and team members meet to review potentially shippable product increment and plan its next steps.

In each Scrum, a team is cross-functional, where everyone is needed to take responsibility from idea to implementation; however, the Scrum teams are supported by two specific roles:

* A Scrum Master: Tasked with coaching the team and assisting team members to use the scrum process to perform at the highest level.
* The Product Owner (PO), on the other hand, represents the business, customers or users, who will guide the team towards building the right product.

Each component within the framework is supposed to serve a specific purpose and is essential to Scrum’s success and usage.

* In Scrum, the events take place as prescribed to maintain regularity.
* In all time-boxed events, each event is given a maximum duration.

***Sprint Terminologies***

***The Product Backlog***

An orderly list of requirements that a scrum team maintains for a product is known as product backlog. Additions to a backlog are commonly written in a story format.

***Sprint backlog***

The sprint backlog is the list of work that has to be finished by the development team during the next sprint.

***Sprint burn-down chart***

A Scrum Master uses the sprint burn-down chart to monitor daily progress of the development team. The sprint burn-down chart shows the remaining work in the sprint backlog.

***Scrum Methodology***

* The new sprint starts immediately after the previous sprint is concluded.
* Every next sprint consists of the sprint planning, daily scrums, the development work, the sprint review, and the sprint retrospective.
* Sprint planning is collaboratively done by the whole Scrum Team.
* A 15-minute time-boxed event is a regular feature of scum, held to create a plan for that day and synchronize the team activities.
* A sprint review is held at the end of the Sprint to inspect the increment and make changes to the Product Backlog, if needed.
* The Sprint Retrospective takes place after the Sprint Review and prior to the next Sprint Planning, during which time the Scrum Team inspects itself and creates a plan for improvements to be enacted during the next Sprint.

***Daily Scrum Meetings***

* The daily scrum meeting also referred to as a “Daily Stand-up Meeting” is a 15-minute meeting for the team, conducted daily to quickly understand the work since the last Daily Scrum Meeting and to create a plan for the next 24 hours.
* The daily scrum meeting is held at the same time and place every day to reduce complexity.

During the meeting, each team member explains:

* + What he/she did yesterday that helped the team meet the Sprint Goal?
  + What he/she will do today to help the Team meet the Sprint Goals?
  + Does he/she see any impediments that prevent he/she or the team from meeting the Sprint Goal?

#### **Traditional SDLC Model Vs Agile Model**

|  |  |
| --- | --- |
| **Agile Model** | **Traditional Model** |
| It is based on the adaptive software development methodologies. | The traditional SDLC models, like the waterfall model, are based on predictive approach. |
| Since the Model follows an Adaptive Approach, there is no detailed planning and clarity on future tasks. Instead, there is only knowledge of what features need to be developed. | This SDLC model usually works with detailed planning, having a complete forecast of the exact tasks and features to be delivered in the next few months. |
| Customer interaction serves as the backbone for the success of Agile technologies. | User feedback generally recorded after the product has been developed. |
| It has open communication with minimum documentation. | It includes a detailed document that all team members are supposed to follow. |

***References*:**

1. <https://www.tutorialspoint.com/software_engineering/software_design_basics.htm>
2. <https://www.techopedia.com/definition/25976/software-development-life-cycle-model->
3. <https://www.tutorialspoint.com/sdlc>
4. <http://ecomputernotes.com/software-engineering/feasibilitystudy>
5. <http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>
6. <http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/sdlc>
7. <http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use>
8. <https://www.tutorialspoint.com/software_engineering/software_testing_overview.htm>

# Module 4

## **Introduction to Web Programming**

## *How did Web Programming begin?*

## **The Internet**

The internet has revolutionized the world of ‘computers and communications’ through its broadcasting and information dissemination capabilities that is quite independent of geographic location of the user.

The discovery of the internet began under the US Department of Defense's Advanced Research Projects Agency (ARPA) net. By the end of the 1970s, a computer scientist named Vinton Cerf had developed a way for computers working on mini-packet-switched networks to communicate with one another. He called his invention “Transmission Control Protocol,” or TCP. Later, he added an additional protocol, known as “Internet Protocol,” which is referred to as TCP/IP.

Due to Cerf’s protocol, the Internet was transformed into a worldwide network. Throughout the 1980s, researchers and scientists were making use of the internet to send files and data from one computer to another. The internet was opened to commercial interests in the late 80s.

However, in 1989 the Internet changed again. That year, a computer programmer, Tim Berners-Lee working in Switzerland at CERN, European Nuclear Research Organization (Conseil Européen pour la Recherche, in French) introduced the World Wide Web ― an Internet that was not simply a way to send files from one place to another but was itself a “web” of information that anyone connected to the Internet could retrieve. Berners-Lee’s Internet was the one that we know today. In 1991 he also went on to build a web browser which was later called *Nexus*.

## **Browsers**

Since then, there have been many different browsers developed for the Internet. In 1992, a group of students and researchers at the University of Illinois developed a sophisticated browser that they called Mosaic (it later became Netscape in 1994). Mosaic offered a user-friendly way to search the web, allowing the public to see words and pictures on the same page and also navigate these pages easily using scrollbars and clickable links. The first version of IE Internet Explorer, (at that time named Microsoft Internet) was first released in 1995 as part of the add-on package for Windows 95.

A series of IEs were later released; however, Netscape had become IE’s major competitor with the launch of Firefox in 2004. Built by Netscape’s Navigator spin-off, Mozilla, Firefox started changing the landscape of web browsers into the competitive space we now know today. Open-source at its core, Firefox was an answer to both those who found Internet Explorer imperfect and those who feared its level of privacy. The launch of Chrome in 2008 marked Google’s entrance into web browsers. Google now releases the majority of Chrome’s source code in an open-source project named “Chromium.”

#### **Search Engines**

A search engine is a program that searches the Internet and finds web pages for the user based on the keywords that you submit. It consists of the following components:

Search engine software with Boolean operators, search fields, display format, etc.

* Spider software
* A database, or a directory
* Algorithms that rank results for relevancy

|  |  |
| --- | --- |
| **Browser** | **Search Engine** |
| A browser is a software program installed locally to your computer, providing access to websites and web pages | The role of a search engine is to search for particular information |
| In a browser window, you type the name of the browser; without a browser, a search engine cannot be opened. | Search Engines use different algorithms for searching information |
| I.E., Firefox, Safari, and Chrome are the most popular web browsers | Google and Yahoo are the most popular search engines. |

## **Social Networking Sites**

#### **Orkut and Facebook**

The very first thing that comes to one’s mind with the word ‘Social Network’ is no doubt Mark Zuckerberg’s ‘Facebook’. However, a few years earlier, the same word would bring to mind *Orkut*. *Orkut* came into existence on January 24th, 2004. *Orkut* was later shutdown in September 2014 by Google in the wake of its new networks, Facebook and Twitter, moving ahead in the social media race.

Social Networking Websites offer a unique platform to connect old friends and people who share common interests. With over 350 million users worldwide, networking via social media (Facebook/Twitter/LinkedIn etc.) has definitely revolutionized the way people are using the internet today.

The internet has not only become an integral part of our personal and social existence, but now our professional world is also dependent upon it as well. Though Facebook started as a social networking site, it is now helping people to enhance their professional scene through personal connections.

## **Smartphones and Tablets**

A smartphone is a mobile phone that performs many of the functions of a computer, has a touch-screen interface, Internet access, and an operating system capable of running downloaded apps.

Prior to the invention of smartphones, there were several devices that were used such as regular mobile phones, and PDA (personal digital assistants) devices. Eventually both technologies were combined and IBM came up with a prototype mobile phone that incorporated PDA features.

The most commonly used OS for smartphones these days is ANDROID. Android is a ‘software bunch’ that comprises not only an operating system, but also middleware and key applications. Released in September 2008, Android uses an operating system based on the Linux kernel. It was designed primarily for touch screen mobile devices, with variants for television, cars and wrist wear. The operating system uses a virtual keyboard, as well as touch inputs that correspond to real-world actions, like swiping, tapping etc.

Android applications are written in java programming language. Android’s java language is available as open source for developers, allowing others to develop applications that can be further used for selling in the android market.

***Tablet***

An Android tablet is a touch-screen, mobile device that runs some version of the Android operating system on it. While it is not a smart phone, with the right software and hardware, you might be able to make calls over Wi-Fi networks.

With a couple work apps and shortcuts, a tablet can be used as a more productive, secondary device for keeping up with email, notes, or other work-related tasks you do on your laptop.

## **Web Development & Programming**

Web development means building a website and deploying it on a web server. Web development requires the use of special programming languages known as scripting languages both at the server end as well as at the client’s end.

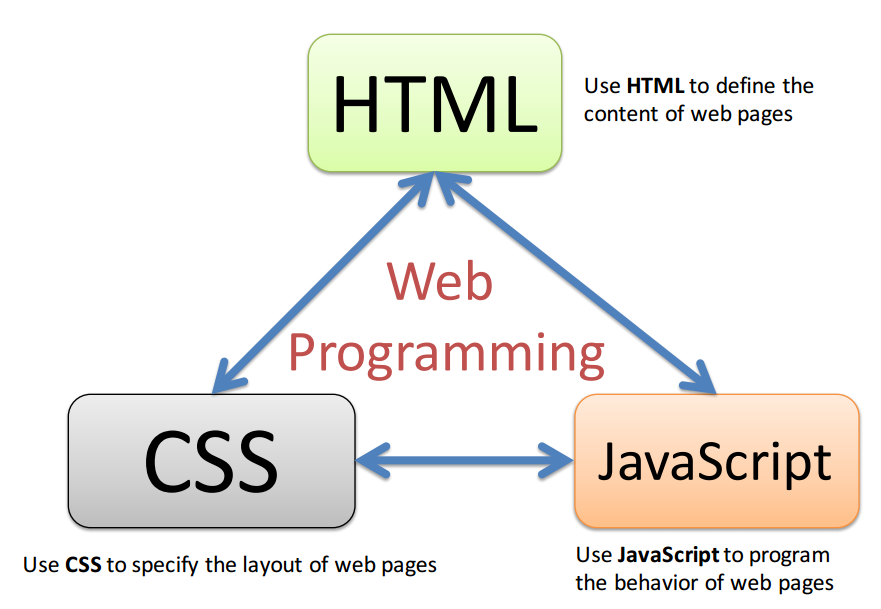
***What is a Web Server?***

A Web Server is a program that serves web pages and web sites to the users. It uses HTTP (Hypertext Transfer Protocol) to serve the files or Web pages to users, in response to their requests, sent through computers' browser-based clients. The server identifies the website through its URL (the web site address) and displays it for the user.

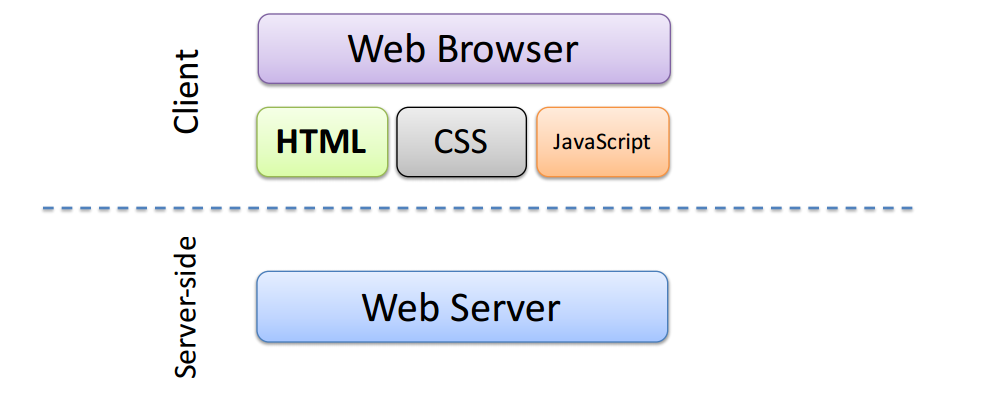
***What is Web Programming?***

Web programming is the whole process of writing, markup, and coding involved in web development to include web content, web client and server scripting, and network security. XML, HTML, JavaScript, Perl 5 and PHP are the most common languages used for Web programming.

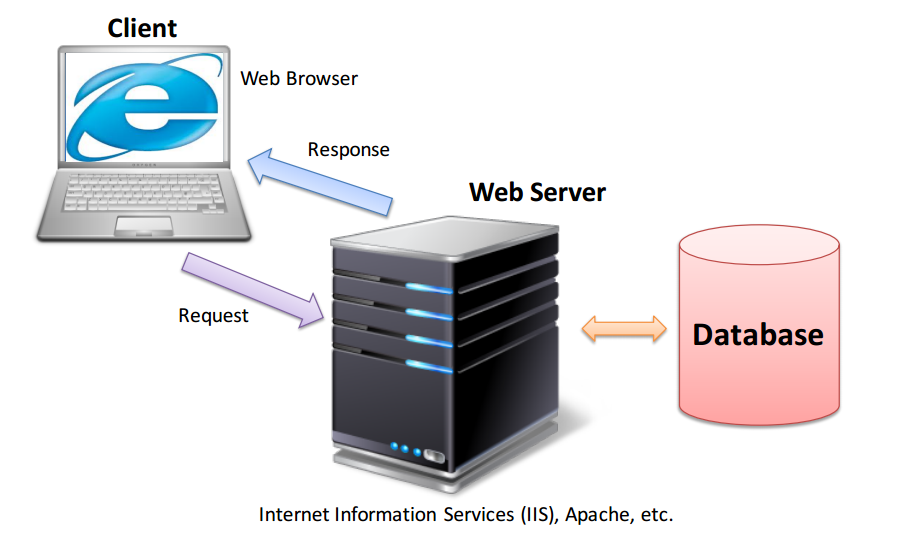
***The Web Programming Triangle***



***Web Architecture***



***Client-Server Example***



## **Concepts of HTML**

HTML (Hypertext Markup Language) is the set of markup symbols or codes to be inserted in a file intended for display on a World Wide Web browser page.

Markup means information that is added to a document to enhance its meaning in certain ways. Markup identifies the different parts of the document through demarcation and labeling through a set of symbols. It also defines how different parts of the document relate to each other.

It is the markup that tells the Web Browser how the words and images of a web page will be displayed for the user. Each individual markup code is referred to as an element, which can also be called as a tag. Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

Recommendations by the World Wide Web Consortium (W3C) are generally adhered to by the major browsers such as Microsoft's Internet Explorer and Netscape's Navigator. However, some additional non-standard codes can also be used in some cases.

HTML 5 is the latest version of HTML.

***A Basic HTML Document***

***HTML – Elements***

* An HTML element is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash.
* HTML documents consist of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

***Example***

<p> is starting tag of a paragraph and </p> is closing tag of the same

<p>This is paragraph</p>

The above pair of tags is an HTML element

***HTML - Attributes***

An attribute is used for defining the characteristics of an HTML element and is placed inside the element's opening tag.

All attributes are made up of two parts: a name and a value.

***Name***

The name is the property you want to define. For example, the paragraph <p> element in the example carries an attribute whose name is aligned, which you can use to indicate the alignment of the paragraph on the page.

***Value***

The value of the property you want to give to the name: Left, Right, Center.

The value is put within quotations.

Example:

<p align="left">This is left aligned</p>

## **How is HTML written?**

<!DOCTYPE html>

<html>

<head>

<title>This is document title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>Document content goes here.....</p>

</body>

</html>

***Some Basic Tags***

***Heading Tags***: <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>

***Paragraph Tag :*** For each paragraph, text should go in between an opening <p> and a closing </p> tag.

***Line Break Tag:*** <br />

<center>

* Puts any content in the center of the page or puts the content of a table cell in the center

The <hr /> tag

* Creates a line from the current position in the document to the right margin and breaks the line accordingly

<pre> tag and the closing </pre> tag

* Any text between the above tags will preserve the formatting of the source document.

&nbsp; tag

* If you do not want the client browser to break the text, a non-breaking space entity is used with the above instead of a normal space.

## **HTML – Formatting**

<b>...</b> - Bold text

<i>...</i> - Italic text

<u>...</u> - Underlined text

<strike>...</strike> - Strikethrough text

<sup>...</sup> - Superscript text

<sub>...</sub> - Subscript text

***HTML – Images and Tables***

Images are very important to beautify as well as to depict many complex concepts in a simple way on your web page.

* You can insert any image in your web page by using <img> tag.
* The simple syntax to use this tag is

<img src="Image URL" ... attributes-list/>

* The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

The HTML tables are created using the <table> tag in which the

<tr> tag is used to create table rows and

<td> tag is used to create data cells.

***Example of Table in HTML***

<table border="1">

<tr>

<td>Row 1, Column 1</td>

<td>Row 1, Column 2</td>

</tr>

<tr>

<td>Row 2, Column 1</td>

<td>Row 2, Column 2</td>

</tr>

</table>

***HTML – Lists***

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

* <ul> - An unordered list. This will list items using plain bullets.
* <ol> - An ordered list. This will use different schemes of numbers to list your items.
* <dl> - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

***HTML – Text Links***

A web page can contain various links that take you directly to other pages and even specific parts of a given page to which they are linked. The following are a few characteristics of links:

These links are known as hyperlinks.

* Hyperlinks allow visitors to navigate between websites by clicking on them.
* You can create hyperlinks using text or images available on a webpage.
* A link is specified using HTML tag <a>.
* This tag is called the anchor tag and anything between the opening <a> tag and the closing </a> tag becomes part of the link.

The following is the simple syntax to use <a> tag.

<a href="Document URL" ... attributes-list>Link Text</a>

***CSS (Cascading Style Sheets)***

* CSS is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web Documents.
* CSS defines how the HTML elements will be displayed.
* It provides an easy control on style and the layout of multiple web pages all at once.

Most web pages are designed from HTML, which is the standard way to beautify plain web text with fonts, colors, graphics, and hyperlinks (clickable text that transports the user somewhere else). But when websites get really big, the use of HTML becomes difficult. However, the use of CSS makes styling of a large number of web pages easy.

CSS can be seen as a kind of computer dress code which can be changed when needed. CSS can be easily separated from HTML, so that the dress code is easy to find and modify, having an ability to rapidly change the entire look of a web site. It is due to these style sheets that entire web pages can be easily modified.

The characteristic feature of CSS is that it cascades. Each defined style adds to the overall theme, yet, it is possible for the most recent style to override earlier styles. For example, with CSS, we can start by saying we want all of our text 12px (12 units) high, but later we can say we want it to be red, too, and still later, we can tell through CSS that we want one phrase to be in bold or italics, and blue not red.

***Advantages of Cascading Style Sheet - CSS***

As has already been defined, Cascading Style Sheets (CSS) describes how documents will be presented on screens, or in print. Since the consortium was founded in 1994, W3C has actively promoted the use of style sheets on the web. CSS has the following advantages:

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags.

* CSS makes maintaining web pages easier.
* Any number of style properties for a given HTML element can be specified using CSS.
  + Each property has a name and a value, separated by a colon (:).
  + Each property declaration is separated by a semi-colon (;).

***Example***:

<p style="color:green;font-size:24px;">Hello, World!</p>

CSS can be used in three ways in a HTML document:

1. In a separate file (external)
2. At the top of a web page document (internal)
3. Right next to the text it decorates (inline)

***External Style Sheets***

Here's an HTML example where external CSS has been used.

A simple web page is made of tags. Everything must go between the opening and closing <html> tags. The <head> section contains invisible directions called meta information.

In the sample file, the top line is a comment and doesn't do anything.

The <body> section is where all the visible stuff is being put. It tells the web page what background color to use for the body section. Right after that, the h1 part says what we want our largest heading (h1) to be.

External style sheets are separate files full of CSS instructions (with the file extension .css). When any web page includes an external style sheet, its look and feel will be controlled by this CSS file (unless we decide to override a style using one of these next two types).

Now, to include this external CSS file ('stylish.css'), we have to include a link for it within the <head> section of our blank web page. With an external style sheet, the look of an entire website can be changed by changing just one file!

The best thing about using external, style sheets is that they're easy to find, which allows us to 'override' an external style sheet for a specific page.

***Internal Style Sheets***

An internal style sheet may be used if one single page has a unique style. This can be easily done by defining the required styles at the top of the page, in the <head> section.

*Note: An external CSS file is never used for designing a single web page.*

Internal styles are defined within the <style> element, inside the <head> section of an HTML page.

A simple example is given below for internal CSS.

***Inline CSS***

An inline style is used when applying a unique style is to be given for a single element. For using inline styles, the style attribute is applied to the relevant element. The style attribute can have any CSS property.

The example below shows how the color and the left margin of a <h1> element can be changed by using an Inline CSS.

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

## **JavaScript**

JavaScript is a programming language through which complex things on web pages are introduced such as regular content updates, interactive maps, animated 2D/3D graphics, videos etc. Java script is the third layer of standard web technologies, after HTML and CSS.

1. **HTML(Structure Layer):** The layer which creates a platform for a website to build upon it.
2. **CSS (Style Layer):** The layer which dictates how a structured HTML document would be displayed in a web browser.
3. **Java (Behavior Layer):** The layer of web page that responds to different user actions or makes changes to a page based on a set of conditions.

#### **Characteristics of JavaScript**

* JavaScript is a lightweight, interpreted programming language which is designed for creating network-centric applications.
* It is complementary to both Java and HTML and can be easily integrated to an open or cross-platform.
* JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page

***Example***

<script language="javascript" type="text/javascript">

JavaScript code

</script>

***First JavaScript Code***

<html>

<body>

<script type="text/javascript">

document.write ("Hello World!")

</script>

</body>

</html>

***Example of JavaScript*** in <head>...</head> Section

<html>

<head>

<script type="text/javascript">

function sayHello() {

alert("Hello World") }

</script>

</head>

<body>

Click here for the result

<input type="button" onclick="sayHello()" value="Say Hello" />

</body>

</html>

***Client-side JavaScript***

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

Characteristics of Client-Side JavaScript

* When the user submits the form, the JavaScript code gets executed, and when all the entries are valid, it is submitted to the Web Server.
* All user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly are initiated through JavaScript.

***Advantages of JavaScript***

1. **Less server interaction**

The user input is validated before the page is sent to the server saving server traffic and load.

1. **Immediate feedback to the visitors**

Incoming visitors don't have to wait for a page to get reloaded to see if they have forgotten to write an entry

1. **Increased interactivity**

JavaScript creates interfaces that react when the user activates them via the keyboard hovering over them with a mouse.

1. **Richer interfaces**

JavaScript provides a rich interface to the visitors such as drag-and-drop components and sliders.

1. **Security**

Client-side JavaScript does not allow the reading or writing of files of the customers for security reasons.

***Limitations of JavaScript***

JavaScript cannot be treated as a full-fledged programming language. It lacks the following important features:

* It cannot be used for networking applications on its own, because there is no such support available in the JavaScript.
* It doesn't have any multithreading or multiprocessor capabilities.

***Events***

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. Events are a part of the Document Object Model (DOM) Level 3. Every HTML element contains a set of events which can trigger JavaScript Code. The following situations can be called as events:

* When the page loads
* When the user clicks a button
* Pressing any key
* Closing or resizing a window

Developers make use the above events to execute the following JavaScript coded responses:

* Causing buttons to close windows
* Displaying messages to users
* Validation of data to be validated
* Any other imaginable response

***Event Types***

* **OnClick Event Type**: occurs when a user clicks the left button of the mouse.
* **OnSubmit Event Type**: occurs when user tries to submit a form.
* **On mouseover**: Occurs when a user hovers the mouse over any element.
* **On mouseout**: Occurs when a user moves the mouse out from that element.

***JavaScript Cookies***

Cookies are data (5 variable-length fields) that gets stored in small text files on your computer. A web server sends a web page to a browser on request, but after the connection is shut down, the server forgets everything about the user. However, through Cookies, the browser is able to remember information about the user.

JavaScript can create, read, and delete cookies with the document.cookie property.

A cookie can be created with JavaScript, as follows:

* document.cookie = "username=John Doe";

You can also add an expiry date (in UTC time).

* document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

The date tells when the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.

With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

* document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

This may be blank if you want to retrieve the cookie from any directory or page.

**Secure**: If this field contains the word "secure,” then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists. Cookies are set and retrieved in the form of key-value pairs. JavaScript can read, create, modify, and delete the cookies that apply to the current web page.

A cookie can be created simply by assigning a string value to the document. The cookie object looks like this:

* document.cookie = "key1=value1;key2=value2;expires=date";

Reading a cookie is just as simple as writing one, because the value of the document.cookie object is the cookie.

To delete a cookie, you just need to set the expiry date to a time in the past.

**References**

1. <http://www.history.com/topics/inventions/invention-of-the-Internet>
2. <http://www.differencebetween.net/technology/internet/difference-between-search-engine-and-browser/#ixzz4wblsdEJR>
3. <https://www.thesnugg.com/a-brief-history-of-smartphones.aspx>
4. <https://www.engineersgarage.com/articles/what-is-android-introduction>
5. <https://www.thesnugg.com/a-brief-history-of-smartphones.aspx>
6. <https://www.engineersgarage.com/articles/what-is-android-introduction>
7. <https://lifehacker.com/top-10-worthwhile-uses-for-tablets-1480435110>
8. <http://searchmicroservices.techtarget.com/definition/HTML-Hypertext-Markup-Language>
9. <http://study.com/academy/lesson/cascading-style-sheets-css-definition-types-examples.html>
10. <https://www.w3schools.com/css/css_howto.asp>
11. <https://www.w3schools.com/js/js_cookies.asp>

# Module 5

## **XML – An Overview**

## **Overview of XML**

XML stands for Extensible Markup Language. Markup, as you know, is information that is added to a document to enhance its meaning in certain ways.

Markup identifies the different parts of the document, demarcates them, and labels them through a set of symbols. It also defines how different parts of the document relate to each other.

## **Main Characteristics of XML**

* It is a text-based markup language derived from Standard Generalized Markup Language (SGML).
* XML has been designed to store, organize, and transport data.
* XML has been designed to be both human and machine-readable.

There are three important characteristics of XML that make it useful in a variety of systems and solutions:

1. **XML is Extensible**

It means that XML allows creating self-descriptive tags, or language, which can be adapted to suit different applications.

1. **XML Carries and Stores the Data**

XML allows storing of data irrespective of how it will be presented.

1. **XML is a Public Standard**

XML is available as an open standard developed by an organization known as the World Wide Web Consortium (W3C).

#### **Advantages of Using XML**

* For large websites, it can work behind the scenes in simplifying the creation of HTML documents.
* XML facilitates the exchange of information between organizations and systems.
* It can be used for offloading and reloading of databases.
* Since XML is suitable for storing and arranging data, it can be used for customizing different data handling needs.
* Virtually any type of data can be expressed as an XML document.
* XML can easily be merged with style sheets to create almost any desired output.

#### **Difference between HTML and XTML**

|  |  |  |
| --- | --- | --- |
| **Language** | **HTML** | **XTML** |
|  | HTML has been designed to display or present the data | XML has been designed to carry the data |
|  | The focus of HTML is on how data looks | The focus of XTML is on what the data is. |
|  | HTML tags are predefined | XML tags are not predefined |

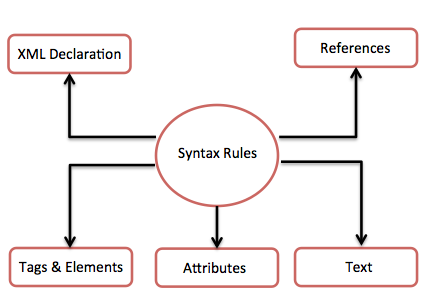
#### **XML Declaration**

It is customary to give an XML declaration in the beginning of the document. An XML document optionally can have only one XML declaration which is written as follows:

<?xml version="1.0" encoding="UTF-8"?>

Where version is the XML version and encoding specifies the character encoding used in the document.

#### **Syntax Rules for XML declaration**

* The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case. 
* The parameter names and values are case-sensitive.
* The names are always in lowercase.
* The order of placing the parameters is important. The correct order is: version, encoding, and standalone.
* If the document contains an XML declaration, then it strictly needs to be the first statement of the XML document.

## **Getting Started with XML**

***1 XML Examples***

***The Basic Example***

The following example shows how XML markup looks when embedded in a piece of text:

<message>

<text>Hello, world!</text>

</message>

Here, the markup symbols, or the tags, are: <message>...</message> and <text>...</text>.

The tags <message> and </message> marks the start and the end of the XML code fragment.

The tags <text> and </text> surround the text Hello, world!

***Simple Example***

<?xml version="1.0"?>

<contact-info>

<name >Tarun Patel</name>

<company>Green Field</company>

<phone>(011) 123-4567</phone>

</contact-info>

You might notice that there are two kinds of information in the above example: The markup, like <contact-info> and the text, or the character data, Green Filed and (011) 123-4567.

## **Tags and Elements of XML**

An XML file is structured by several XML-elements, also called XML-nodes or XML tags. XML-elements' names are enclosed by triangular brackets < > as shown below:

<element>

***What exactly is an XML element?***

XML elements can also be defined as building blocks of an XML document. However, it is important to note that an XML element is everything from (including) the element's start tag to (including) the element's end tag.

Besides being the building block of the XML document, an element can also behave as a container to hold:

1. Text
2. Attributes
3. Other elements
4. Media objects
5. A mix of the above

***Example for XML Elements***

bookstore>

<book category="children">

<title>The Famous Five</title>

<author>Enid Blyton</author>

<year>1943</year>

<price>29.99</price>

</book>

<book category="web">

<title>Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price>39.95</price>

</book>

</bookstore>

<title>, <author>, <year>, and <price> have text content because they contain text (like 29.99).

<bookstore> and <book> have element contents, because they contain elements.

<book> has an attribute (category="children").

***Syntax Rules for Tags and Elements***

Element Syntax:

Each XML-element needs to be closed either with start or with end elements as shown below:

<element>....</element>

or in simple-cases, just this way:

<element/>

## **XML Documents**

***What is an XML Document?***

* An XML document is a basic unit of XML information. It is composed of elements and other markup in an orderly package.
* An XML document can contain a wide variety of data
* A database of numbers, numbers representing molecular structure, or a mathematical equation are a few examples of XML data.

***A simple example:***

<?xml version="1.0"?>

<contact-info>

<name> Tarun Patel </name>

<company>Green Field</company>

<phone>(011) 123-4567</phone>

</contact-info>

***Rules for XML Documents***

#### **XML Documents Must Have a Root Element**

An XML document can have only one root element. For example, the following is not a correct XML document, because both the x and y elements occur at the top level without a root element:

|  |  |  |  |
| --- | --- | --- | --- |
| **Wrong** | | **Right** | |
|  | <x>...</x>  <y>...</y> | | <root>  <x>...</x>  <y>...</y>  </root> | |

The example below shows a correctly formed XML document:

***Example***

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

#### **XML Elements Must be Properly Nested**

An XML element can contain multiple XML elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

|  |  |
| --- | --- |
| **Incorrect Nested Tags:** | **Correct Nested Tags:** |
| <?xml version="1.0"?>  <contact-info>  <company>Green Filed  <contact-info>  </company> | <?xml version="1.0"?>  <contact-info>  <company>Green Field</company>  <contact-info> |

#### **The Names of XML-elements are Case-sensitive.**

That means the name of the start and the end elements needs to be in the same case.

For example, <contact-info> is different from <Contact-Info>.

(Ex. <address> )This is the wrong syntax </Address>

The correct way is to we use the same case to name the start and the end tag.

(Ex. <address> ) This is the correct syntax</address>

#### **XML Attribute Values must be Quoted**

We must first know the properties of an Attribute:

1. Attributes are designed to contain data related to a specific element.
2. An XML element can have one or more attributes.
3. An Attribute gives more information about XML elements by defining properties of elements.
4. An attribute specifies a single property for the element, using a name/value pair.
5. The name is followed by an = sign and a string value which is kept inside double(" ") or single(' ') quotes.

***Syntax to write an XML Element with attributes***:

<element-name attribute1 attribute2>

....content

</element-name>

Where element-name is the name of the element, the name in the start and end tags must match.

Attribute1 and attribute2 are attributes of the element separated by white spaces.

***Empty Element***

An element with no content is said to be empty.

In XML, you can indicate an empty element as follows:

**<element></element>**

An empty element (element with no content) has the following syntax:

**<name attribute1 attribute2.../>**

The scope of elements is delimited by start and end tags. For empty elements, it is delimited by an empty-element tag.

***Element VS Attribute***

|  |  |
| --- | --- |
| **Element** | **Attributes** |
| <person>   <gender>female</gender>   <firstname>Anna</firstname>   <lastname>Smith</lastname> </person>  In the above example, gender is an element. | <person gender="female">   <firstname>Anna</firstname>   <lastname>Smith</lastname> </person>  In the above example gender is an attribute. |
| Elements can contain multiple values. | Attributes cannot contain multiple values |
| Elements can contain tree structures. | Attributes cannot contain tree structures |
| Elements are easily expandable for future changes | Attributes are not easily expandable for future changes. |

***A simple example for the use of an Attribute***

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE garden [

XML Attributes

<!ELEMENT garden (plants)\*>

<!ELEMENT plants (#PCDATA)>

<!ATTLIST plants category CDATA #REQUIRED>

]>

<garden>

<plants category="flowers" />

<plants category="shrubs">

</plants>

</garden>

Attributes are used for distinguishing elements of the same name, when a new element needs to be created for every situation.

In the above example for distinguishing the plant names, the attribute ‘category’ has been used which helps in assigning different values to each of the elements. Hence, we have two categories of plants: one flowers and the other, color. Therefore, we have two plant elements with different attributes.

You can also observe that the attribute is defined at the beginning of XML.

## **XML Declaration**

As specified earlier, XML declaration is always the first statement in the document. XML declaration is optional but when present, it contains details that prepare an XML processor to break the XML document into its component <?xml version="1.0" encoding="UTF-8"?>

Either single or double quotes may be used for attributes.

#### **Following syntax shows XML declaration:**

<?xml

version="version\_number"

encoding="encoding\_declaration"

standalone="standalone\_status"

?>nt parts and describe their syntactic roles. If the XML declaration is included, it must contain a version number attribute followed by an encoding declaration.

## **Document Type Declaration**

* The XML Document Type Declaration, generally known as DTD, is a precise way to describe XML language.
* DTD has a very important function to perform the following jobs in the XML document:
  + Check the vocabulary and validity of the XML document structure against grammatical rules of that particular XML language.
* DTD always starts with <!DOCTYPE delimiter or identifier which may be the path to a file on the system or URL to a file on the internet.

**External VS Internal DTD**

An XML DTD can be either specified inside the document, or it can be kept in a separate document and then linked to the document.

***Internal Declarations or internal DTD***

As the name specifies, these are declared within the XML files. Here the standalone attribute in XML declaration must be set to ‘yes’. This means, the declaration works independent of any external source.

The square brackets [ ] enclose an optional list of entity declarations called Internal Subset.

Remember the following two rules if it is supposed to be an internal DTD:

1. Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.
2. The ‘Name’ in the document type declaration must match the element type of the root element.

***XML DTD – Example***

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<!DOCTYPE address [

<!ELEMENT address (name,company,phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

]>

<address>

<name>Tarun Patel</name>

<company>Green Field</company>

<phone>(011) 123-4567</phone>

***External DTD***

If the DTD is pointing to an external path, it is called an External Subset. It is declared outside the XML file. External DTD is accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as an external DTD, a standalone attribute in the XML declaration must be set to ‘no’. This means that the XML declaration includes information from the external source.

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!ELEMENT address (name,company,phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

<!DOCTYPE address SYSTEM "address.dtd">

<address>

<name> Tarun Patel </name>

<company>Green Field</company>

<phone>(011) 123-4567</phone>

</address>

## **XML Tree Structure**

* An XML document is always descriptive.
* An XML Tree plays an important role in easily describing any XML document through its structure.
* The tree structure contains root (parent) elements, child elements and so-on.
* The tree structure starts at the root and goes towards all succeeding branches and sub-branches.
* The parsing starts at the root, then moves down the first branch to an element. Next, it takes the first branch and so-on to the leaf nodes.

***XML Tree Structure – Example***

<?xml version="1.0"?>

<Company>

<Employee>

<FirstName>Tarun</FirstName>

<LastName>Patel</LastName>

<ContactNo>1234567890</ContactNo>

<Email>tarunpatel@greenfields.com</Email>

<Address>

<City>New Delhi</City>

<State>Delhi</State>

<Zip>11019</Zip>

</Address>

</Employee>

</Company>

***XML Validation***

The process through which an XML document is validated is known as XML validation.

* An XML document is said to be valid if its contents match with its:
* Elements
* Attributes
* Associated document type declaration (DTD)
* The document also complies with the constraints expressed in it.

***Validation is dealt in two ways by the XML parser***:

1. Well-formed XML document
2. Valid XML document

***Well-formed XML Documents***

Well-formed XML documents are those which have:

* The correct syntax
* All XML elements have a closing tag
* All tags in the document follow the correct ordering of the tags, i.e., the inner tag must be closed before closing the outer tag
* XML tags that are case sensitive
* XML elements are properly nested
* A root element

An XML document is said to be well-formed if it also adheres to the following rules:

* Non-DTD XML files must use the predefined character entities for amp(&), apos(single quote), gt(>), lt(<), quot(double quote).
* It must have only one attribute in a start tag, which needs to be quoted.

***Example of well-formed XML document***

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<!DOCTYPE address

[

<!ELEMENT address (name,company,phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

]>

<address>

<name>Tarun Patel</name>

<company>Green Fields</company>

<phone>(011) 123-4567</phone>

</address>

***What is a Valid XML Document?***

If an XML document is well-formed and also has an associated Document Type Declaration (DTD), then it can be called as a valid XML document.

As already discussed, two different document type definitions can be used for a Valid XML document:

1. DTD - The original Document Type Definition
2. XML Schema - An XML-based alternative to DTD

A document type definition defines the rules, legal elements, and attributes for an XML document.

**XML Schemas**

An XML Schema, commonly known as XML Schema Definition (XSD), is an XML-based alternative to DTD which performs the function of validating the XML document. It has the following characteristics:

* XML Schema is used for describing and validating the structure and the content of XML data, similar to a database schema that describes the data in a database
* XML schema defines the elements, attributes and data types
* Schema element supports namespaces

***An XML schema in a document is declared as follows:***

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

XML Schemas – Example

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="contact">

<xs:complexType>

<xs:sequence>

<xs:element name="name" type="xs:string" />

<xs:element name="company" type="xs:string" />

<xs:element name="phone" type="xs:int" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

***Why XML Schemas are more powerful than DTD?***

* They are written in XML
* They are extensible to additions
* XML Schemas support data types
* XML Schemas support namespaces

***Why Use an XML Schema?***

* With XML Schema, the XML files can be made to carry a description of its own format.
* Independent groups of people can agree on a standard for interchanging data through the use of XML Schema.
* Verification of data can be easily facilitated through the use of XML Schema.

***XML Namespaces***

XML Namespaces provide a method to avoid element name conflicts. In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications; an example of this would be an xml document carrying HTML information about a table’s width height etc. and another XML document containing information about a coffee table’s width, height etc. This problem is solved by adding a prefix such as <h:table> and <f:table> which can be defined by XML schemas or by XMLS in an XML document as shown in the box.

A Namespace can also be defined as a set of unique names. Namespace is a mechanism by which element and attribute names can be assigned to a group. The Namespaces are identified by URI (Uniform Resource Identifiers).

***Namespace Declaration***

A Namespace is declared using reserved attributes. Such an attribute name must either include xmlns or begin with xmlns shown in the figure below:

<element xmlns:name = "URL">

Syntax:

* The Namespace starts with the keyword xmlns
* The word name is the Namespace prefix
* The URL is the Namespace identifier

Namespace affects only a limited area in the document. An element containing the declaration and its entire lineage comes in the scope of the Namespace.

The Namespace prefix used here is cont, and the Namespace identifier (URI) as www.greenfields.com/profile.

This means, the element names and attribute names with the cont prefix (including the contact element), all belong to the www.greenfields.com/profile namespace.

***Following is a simple example of XML Namespace***

<?xml version = "1.0" encoding = "UTF-8"?>

<cont:contact xmlns:cont = "www.greenfields.com/profile">

<cont:name>Tarun Patel</cont:name>

<cont:company>Green Fields</cont:company>

<cont:phone>(011) 123-4567</cont:phone>

</cont:contact>

***References***:

1. <https://www.w3schools.com/xml/>
2. <https://www.tutorialspoint.com/xml/>
3. <https://www.w3schools.com/xml/xml_namespaces.asp>

# Module 6

## Core Concepts of Relative Database Management Systems (RDBMS)

## **File Management System**

#### **What is Data?**

* Data is easiest defined as known facts having an implied meaning
* It compiles statistics collected for reference or analysis
* Data is organized information
* Data is constituted by quantities, characters, or symbols on which operations are performed by a computer
* Data in a computer is stored inside the files

#### **What is File Management System?**

A File Management System (FMS) is a system that an operating system or program of a computer uses for organizing and keeping track of files. Most computers have a hierarchical file management system that follows a tree structure for directories in which files are organized. Each operating system has its own file management system.

***Drawbacks of File Management System***

* There is separate data for every individual program
* There is always a chance of duplicated data
* Data is not uniformity stored
* Data requires more storage space
* There is no security control over the data

## **Database Management System (DBMS)**

***What is a Database?***

* A database is a collection of logically related data in one place.
* A database is a collection of information or data that is organized in such a way that it can be easily accessed, managed, and updated.
* A structured set of data held in a computer that can be accessed in various ways.

***What is a Database Management System?***

* A Database Management System is a record keeping system managed by a computer that allows you to store, manipulate, and retrieve data easily.
* A DBMS is, in fact, a System Software used for creating and managing databases that are stored in a computer. The DBMS provides both the users and programmers a systematic way to create, retrieve, update, and manage data.
* Almost every computer application needs a database to store its data for retrieval at a later date. Hence, the database is one of the most important components of any application. Many times, the performance of the application is dependent on the design of the database. A poorly designed database can drastically affect the performance of the application.

***Goals of a Database Management System***

* Provides an efficient environment that provides access to data in database
* Added security
* Provides control in concurrency and recovery in the event of crash

***Benefits of Database Approach:***

* Reduction in Redundancy
* Avoids Inconsistency
* Easy Sharing of Data
* Enforcing of Standards
* Takes Care of Security
* Maintains Integrity
* Provides for Data Independency

**Types of Database Models**

***Hierarchical Model***

A hierarchical model is the oldest database model that greatly influenced the database model structure.

In the 1960's, North American Rockwell, in partnership with IBM, developed a database processing language called Data Language (DL) and also produced the first Information Management System (IMS). IMS was the world's leading mainframe hierarchical DBMS of the 70s and 80s.

Similarly, the hierarchical model was the first database model that had succeeded in overcoming the limitations of the traditional file system. Hierarchical databases were predominantly used when the volume of transactions were large, and the transaction operations did not change much over a period of time.

***Network Model***

A network database model is a model that allows multiple records to be linked to a single owner file. The model can be seen as an upside-down tree whose branches include member information linked to the owner (seen at the base of the tree). The multiple links of information allows the network database model to be extremely flexible. The relationship of information in the network database model is defined as a ‘many-to-many relationship’ because one owner file can be linked to many member files and vice-versa.

***Relational Model***

The relational model for a database provides a new approach to managing data using a logical, well-defined, and consistent data structure/language available for storage and retrieval operations. The relational model is also good to use when faced with integrity constraints.

## **What is RDBMS?**

RDBMS stands for Relational Database Management System. RDBMS data is structured through a table, in records, and in fields. RDBMS also provides relational operators to manipulate the stored data. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The Relational database management system (RDBMS) is a database management system (DBMS) that is primarily based on the relational model as introduced by E. F. Codd in 1970.

Codd introduced 12 rules for describing his relational model which are followed by all RDBMS applications.

**Rule 1: Information Rule**

The data stored in a database, be it user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

**Rule 2: Guaranteed Access Rule**

Every single data element (value) is guaranteed to be logically accessible with a combination of table-name, primary-key (row value), and attribute-name values (column value). No other means, such as pointers, can be used to access data.

**Rule 3: Systematic Treatment of NULL Values**

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

**Rule 4: Active Online Catalog**

The structure description of the entire database must be stored in an online catalog, known as the data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog and the database itself.

**Rule 5: Comprehensive Data Sublanguage Rule**

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered a violation.

**Rule 6: View Updating Rule**

All the views of a database, which can theoretically be updated, must also be updatable by the system.

**Rule 7: High-Level Insert, Update, and Delete Rule**

A database must support high-level insertion, updating, and deletion. This must not be limited to a single row; that is, it must also support union, intersection, and minus operations to yield sets of data records.

**Rule 8: Physical Data Independence**

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

**Rule 9: Logical Data Independence**

The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rules to apply.

**Rule 10: Integrity Independence**

A database must be independent of the application that uses it. All of its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

**Rule 11: Distribution Independence**

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

**Rule 12: Non-Subversion Rule**

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

***The Structure of RDBMS***

The data in RDBMS is stored in database objects called tables. The collection of related data entries is organized in the form of a table with columns and rows. Remember, that a table is the simplest and the most common form of data storage in a relational database with each table having its own primary key.

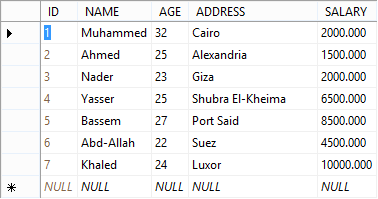
#### **Important RDBMS Terminology**

* **Relation**: A table
* **Tuple**: A row in a table
* **Attribute**: A Column in a table
* **Degree**: Number of attributes
* **Cardinality**: Number of tuples
* **Primary Key**: a unique identifier for the table

*Note: A table can have duplicate tuples, whereas a true relation cannot have duplicate tuples.*

## **RDBMS Characteristics**

1. Every table is broken up into smaller entities called fields. For example, the fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.



1. A field is a column in the table designed to maintain specific information about every record in the table.

A record, also called a row of data, is a horizontal entity representing a set of related data. The record is every individual entry that exists in a table.

**Database Designing**

Database design involves the following 3 steps:

1. Conceptual design: (using ER Model)
2. Fine-tuning of Schema: (Normalization)
3. Physically designing Database and Tuning

## **ER (Entity Relationship) Modeling**

Transforming the ER model (The ER diagram) to a relational model is one of the most essential steps required for developing a database application.

#### **ER Terminology**

* **Entity**: Anything that exists and is distinguishable
* **Attribute**: Properties that describe an entity are its attributes
* **Relationship**: An association between entities

**Table: Converting E-R Model to Relational Model**

|  |  |  |
| --- | --- | --- |
| **ER Model** | **Relational Model** | **Database** |
| Entity | Relation | Table |
| Entity Instance | Tuple | Row |
| Attribute | Attribute | Column |
| Identifier | Key | Key |

#### **Some Definitions in ER**

**Schema**: The design of a database is called the schema. Designing the database is done at three levels: 1) Physical schema 2) Logical schema 3) View schema.

**Instance**: The data stored in a database at a particular moment of time is called the instance of database. A database schema defines the variable declarations in tables belonging to a particular database; so, the value of these variables at a particular moment of time is called the instance.

Different Types of Keys:

* Key
* Super Key
* Candidate Key
  + Primary Key
  + Alternate/Secondary Key

**Candidate Key**: The minimal set of attributes that uniquely identify a tuple is known as a Candidate Key. For example: STUD\_NO in STUDENT relation.

The value of a candidate key is unique and non-null for every tuple.

There can be more than one candidate key in a relation. For Example, STUD\_NO as well as STUD\_PHONE are both candidate keys for the relation STUDENT.

The candidate key can be simple (having only one attribute), as well as composite. For Example, {STUD\_NO, COURSE\_NO} is a composite candidate key for relation STUDENT\_COURSE.

**Super Key**: The set of attributes that can uniquely identify a tuple is known as a Super Key. An example of this is STUD\_NO, (STUD\_NO, STUD\_NAME) etc.

A super key is formed by adding a zero or more attributes to a candidate key.

A candidate key is always a super key but this is not true vice-versa.

**Primary Key**: There can be more than one candidate key in a relation, out of which, one can be chosen as the primary key. For Example, STUD\_NO as well as STUD\_PHONE are both candidate keys for relation STUDENT, but STUD\_NO can be chosen as the primary key (only one out of many candidate keys).

**Alternate Key**: The candidate key other than the primary key is known as an alternate key. In the example provided above, STUD\_PHONE would be the alternate key (only one key is chosen as alternate key out of many candidate keys).

**Foreign Key**: When an attribute takes the values that are present as values of another attribute, it will be a foreign key to the attribute to which it refers.

* The relation which is being referenced is called referenced relation and the corresponding attribute is known as a referenced attribute.
* The relation that refers to a referenced relation is called referencing relation and its corresponding attribute is called referencing attribute.
* The referenced attribute of a referencing attribute should be the primary key. For Example, STUD\_NO in STUDENT\_COURSE is a foreign key to STUD\_NO in STUDENT relation.

## **Normalization**

Database normalization is the process of efficiently organizing data in a database. There are two reasons for the normalization process:

* Eliminating redundant data, e.g. the same data that exists in more than one table. Ensuring sensible data dependencies in related data.
* Overcoming anomalies that can occur because of insertion, deletion, and updating.

If data is logically stored, the amount of space a database consumes will almost always be reduced. Normalization consists of a series of guidelines that help assist you in creating a good database structure.

#### **Normalization in short, is a logical multi-step process that removes duplicated data.**

Normalization guidelines are divided into a number of normal forms; these forms serve to format how the database structure is to be laid out. The aim of normal forms is to organize the database structure so that it complies sequentially with the rules of first normal form, second normal form, and finally with third normal form.

You can also choose to take it further by going to fourth normal form, fifth normal form, and so-on, but in general, third normal form is sufficient enough.

#### **Normal Forms:**

#### **First Normal Form (1NF)**

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only the atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_id** | **emp\_name** | **emp\_address** | **emp\_mobile** |
| 101 | Harish | New Delhi | 8912312390 |
| 102 | Janardan | Kanpur | 8812121212 9900012222 |
| 103 | Ramnath | Chennai | 7778881212 |
| 104 | Laxman | Bangalore | 9990000123  8123450987 |

Two employees (Janardann & Laxmanr) each have two mobile numbers that the company stored in the same field (see figure above).

This table is not in 1NF as the rule says “each attribute of a table must have an atomic (single) value.” The emp\_mobile values for employees Janardan & Laxman violates that rule.

To make the table comply with 1NF we should have the data like this:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_id** | **emp\_name** | **emp\_address** | **emp\_mobile** |
| 101 | Harish | New Delhi | 8912312390 |
| 102 | Janardan | Kanpur | 8812121212 |
| 102 | Janardan | Kanpur | 9900012222 |
| 103 | Ramnath | Chennai | 7778881212 |
| 104 | Laxman | Bangalore | 9990000123 |
| 104 | Laxman | Bangalore | 8123450987 |

#### **Second Normal Form (2NF)**

A table is said to be in 2NF if both the following conditions apply

1. Table is in 1NF (First normal form).
2. No non-prime attribute is dependent on the proper subset of any candidate key of the table.

An attribute that is not part of any candidate key is known as a non-prime attribute.

**Example**: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like the figure below. Since a teacher can teach more than one subject, the table can have multiple rows for the same teacher.

|  |  |  |
| --- | --- | --- |
| **teacher\_id** | **Subject** | **teacher\_age** |
| 111 | Math | 38 |
| 111 | Physics | 38 |
| 222 | Biology | 38 |
| 333 | Physics | 40 |
| 333 | Chemistry | 40 |

***Candidate Keys***:

{teacher\_id, subject}

Here the teacher\_age is a non-prime attribute.

The table is in 1NF because each attribute has an atomic value. However, it is not in 2NF because the non-prime attribute teacher\_age is dependent on teacher\_id alone (which is a subset of the candidate key). This violates the rule for 2NF which says “no non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table comply with 2NF we can break it into two tables like this:

teacher\_details table:

|  |  |
| --- | --- |
| **teacher\_id** | **teacher\_age** |
| 111 | 38 |
| 222 | 38 |
| 333 | 40 |

**teacher\_subject table:**

|  |  |
| --- | --- |
| **teacher\_id** | **Subject** |
| 111 | Math |
| 111 | Physics |
| 222 | Biology |
| 333 | Physics |
| 333 | Chemistry |

Now both of these tables comply with Second normal form (2NF).

#### **Third Normal Form (3NF)**

A table design is said to be in 3NF if both the following conditions apply:

1. Table is in 2NF.
2. The Transitive Functional Dependency of Non-Prime Attributes on any super key is removed.

An attribute that is not part of any candidate key is known as a non-prime attribute.

In other words, a table is in 3NF if it is in 2NF and for each functional dependency X->Y at least one of the following conditions hold good:

1. X is a super key of table
2. Y is a prime attribute of table

An attribute that is a part of one of the Candidate keys is known as a Prime Attribute.

**Example**: Suppose a company wants to store the complete address of each employee. They create a table named “Employee Details” similar to the figure below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **emp\_id** | **emp\_name** | **emp\_zip** | **emp\_state** | **emp\_city** | **emp\_district** |
| 1001 | Suneel | 282005 | UP | Agra | Dayal Bagh |
| 1004 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Laxman | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Roshan | 292008 | UK | Pauri | Bhagwan |
| 1201 | Shankar | 222999 | MP | Gwalior | Ratan |
|  |  |  |  |  |  |

***Super keys***: {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip} …so on

***Candidate Keys:*** {emp\_id}

***Non-prime attributes***: All attributes except emp\_id are non-prime, as they are not part of any candidate keys.

Here, emp\_state, emp\_city & emp\_district are dependent on emp\_zip. And, emp\_zip is dependent on emp\_id, which makes the non-prime attributes (emp\_state, emp\_city & emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table comply with 3NF, we need to break the table into two tables so that the transitive dependency is removed.

#### **Employee Table:**

|  |  |  |
| --- | --- | --- |
| **emp\_id** | **emp\_name** | **emp\_zip** |
| 1001 | Suneel | 282005 |
| 1004 | Ajeet | 222008 |
| 1006 | Laxman | 282007 |
| 1101 | Roshan | 292008 |
| 1201 | Shankar | 222999 |

#### **Employee\_zip table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **emp\_zip** | **emp\_state** | **emp\_city** | | **emp\_district** | | |
| 282005 | UP | Agra | | Dayal Bagh | | |
| 222008 | TN | Chennai | | M-City | | |
| 282007 | TN | Chennai | | Urrapakkam | | |
| 292008 | UK | Pauri | | Bhagwan | | |
|  |  | |  | |  |

***References:***

1. <http://a4academics.com/tutorials/24-sql-tutorial/15-introduction>

2. <http://www.computerbusinessresearch.com/Home/database/network-database-model>

3. <http://www.fastlearning.in/sql/sql_rdbms_concepts.php>

4. <https://beginnersbook.com/2015/05/normalization-in-dbms/>

5. <http://searchsqlserver.techtarget.com/definition/database-management-system>

6. <http://holowczak.com/converting-e-r-models-to-relational-models/>

7. <http://www.geeksforgeeks.org/dbms-keys-candidate-super-primary-alternate-and-foreign/>

# Module 7

## SQL Basics

## **Concepts of SQL:**

The World Wide Web today consists of many dynamic websites that are functioning only because of their databases. Database management is a complicated process; however, through the use of SQL languages, it has now become much easier. SQL stands for Structured Query Language (pronounced "ess-que-el") which is the standard computer language that can communicate with a database and has been specially designed for the retrieval and management of data in a relational database.

The SQL programming language was first developed in the 1970s by IBM researchers, Raymond Boyce, and Donald Chamberlin, for IBM's original relational database management systems known as "System R” and had called it as SEQUEL. The idea for developing this special programming language occurred following the publishing of Edgar Frank Todd's paper, "A Relational Model of Data for Large Shared Data Banks," in 1970. Todd had proposed that all data in a database should be represented in the form of relations. It wasn't until several years later, however, that the SQL language was made available publicly. In 1979, a company called Relational Software, which later came to be known as Oracle, commercially released its own version of the SQL language called Oracle V2.

Both the industry-level and academic needs are served by the SQL, which can be efficiently used both for individual computers as well as for corporate servers. Due to the progressive database technology, SQL-based applications have now become easily affordable even for the regular users.

All common relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, Postgres, and the SQL Server make use of SQL elements for performing tasks like updating data, manipulating data, or retrieving data from a database. However, by doing this, they are making use of different dialects as listed in the table below:

|  |  |
| --- | --- |
| **Server** | **SQL** |
| MS SQL | T-SQL |
| Oracle | PL/SQL |
| MS Access | JET SQL (native format) |

## **Where is SQL used?**

SQL is used in the following situations:

1. To access data in relational database management systems
2. Describe the data
3. Define the data in database and manipulate that data
4. Create and drop databases and tables
5. Create view, stored procedures and functions in a database
6. Set permissions on tables, procedures and views
7. SQL facilitates easy embedding of SQL modules, libraries & pre-compilers within other languages

## **SQL Language Elements**

For the convenience of all SQL developers, necessary language commands in the corresponding database management systems are usually executed through a specific SQL command-line interface (CLI). The SQL language comprises the following elements:

**Clauses** - The components of the statements and the queries.

**Expressions** - Used for producing scalar values or tables consisting of columns and rows of data.

**Predicates** – Used for specifying conditions to limit the effects of the statements and the queries, or to change the flow of the program.

**Queries** - Based on a given criterion, the queries are used for retrieving the data.

**Statements** -Used for controlling transactions, program flow, connections, sessions, or diagnostics.

In database systems, queries are sent from a client program through the SQL statements to the server where the databases are stored. In response, the server processes the SQL statements and returns replies to the client program, executing a wide range of data manipulation operations from simple data inputs to complicated queries in fraction of seconds.

**SQL Data Types**

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| VARCHAR2(size) | Variable-length character data |
| CHAR(size) | Fixed-length character data |
| NUMBER(p,s) | Variable-length numeric data |
| DATE | Date and time values |

## **SQL Operators**

An operator in SQL performs the following functions:

1. An operator is a reserved word or a special character used primarily in a SQL statement's WHERE clause.
2. It manipulates individual data items and returns a result. The data items are also called operands or arguments.
3. It performs comparisons and arithmetic operations. For example, the multiplication operator is represented by an asterisk (\*) and the operator that tests for nulls is represented by the keyword IS NULL.
4. Operators are also used for specifying conditions in an SQL statement.
5. Operators serve as conjunctions for multiple conditions in a statement.

## **Classes of Operators**

There are two general classes of operators: Unary and Binary

|  |  |
| --- | --- |
| **Urinary** | A unary operator operates on only one operand. A unary operator typically appears with its operand in this format: operator operand. |
|  |  |
| **Binary** | A binary operator operates on two operands. A binary operator appears with its operands in this format: |
|  | operand1 operator operand2 |

Other operators with special formats accept more than two operands. If an operator is given a null operand, the result is always null. The only operator that does not follow this rule is concatenation (||).

## **Type of Operators**

#### **SQL Arithmetic Operators:**

Assume variable A holds 10 and variable B holds 20:

+ Addition - Adds values on either side of the operator

a + b will give 30

- Subtraction - Subtracts right hand operand from left hand operand

a - b will give -10

\* Multiplication - Multiplies values on either side of the operator

a \* b will give 200

/ Division - Divides left hand operand by right hand operand

b / a will give 2

#### **SQL Comparison Operators**

Assume variable A holds 10 and variable B holds 20:

= Checks if the values of the two operands are equal. If yes, then the condition becomes true.

(a = b) is not true.

!= Checks if the values of the two operands are equal. If the values are not equal, then the condition becomes true.

(a != b) is true.

<> Checks if the values of the two operands are equal or not. If the values are not equal, then the condition becomes true.

(a <> b) is true.

> Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.

(a > b) is not true.

< Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.

(a < b) is true.

>= Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.

(a >= b) is not true

<= Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.

(a <= b) is true

#### **SQL Logical Operators:**

|  |  |  |
| --- | --- | --- |
| Serial Number | **Name of Operator** | **Function** |
| 1 | ALL | Compares value to all values belonging to another value set |
| 2 | AND | Allows multiple conditions in a SQL statement’s WHERE clause |
| 3 | ANY | Compares a value to any applicable value in the list according to the condition |
| 4. | BETWEEN | Searches values that are within a given set of minimum and maximum values |
| 5 | EXISTS | Searches for the presence of a row in a specified table meeting a certain criterion |
| 6 | IN | Compares a value to a list of specified values |
| 7 | LIKE | Compares a value to similar values using wildcard operators |
| 8 | NOT | Reverses the meaning of the logical operator with which it is used (NOT EXISTS, NOT BETWEEN, NOT IN) |
| 9 | OR | Combines multiple conditions in a SQL statement's WHERE clause. |
| 10. | NULL | Compares a value with a NULL value |

## **SQL Statements or Commands**

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE, and DROP. These commands can be classified into groups based on their nature:

***DDL - Data Definition Language :( CREATE, ALTER, DROP)***

***SQL CREATE TABLE***

Creating a basic table also involves naming the table, defining its columns, and also defining the column's data type.

***Syntax***:

CREATE TABLE table\_name ( column1 datatype, column2 datatype, column3 datatype, ..... columnN datatype,

PRIMARY KEY (one or more columns); follows the CREATE TABLE statement.

Then, in brackets, comes the list defining each column in the table and what sort of data type it is. A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement. The new table has the same column definitions. All columns or specific columns can be selected.

When you create a new table using an existing table, the new table would be populated using the existing values in the old table.

***Syntax***:

CREATE TABLE NEW\_TABLE\_NAME AS

SELECT [ column1, column2...columnN ] FROM EXISTING\_TABLE\_NAME

[ WHERE ]

Here, column1 and column2 are the fields of existing table, so the same would be used to create the fields of a new table.

***SQL ALTER TABLE***

This command is used for modifying the definition (structure) of a table by modifying the definition of its columns as follows:

* 1. Add, drop, and modify table columns
  2. Add and drop constraints
  3. Enable and disable constraints

***SQL DROP TABLE***

It deletes an entire table, a view of a table, or other objects in the database.

***DML*** - ***Data Manipulation Language: (INSERT, UPDATE, DELETE).***

***SQL INSERT Query***

The SQL INSERT INTO Statement is used for adding new rows of data to a table in the database.

***Syntax***:

There are two basic syntaxes of INSERT INTO statement:

INSERT INTO TABLE\_NAME (column1, column2, column3,...columnN)]

VALUES (value1, value2, value3,...valueN);

Here, column1, column2, and columnN are the names of the columns in the table that data is to be inserted.

There is no need to specify the column(s) name in the SQL query when values are to be added for all the columns in a table.

However, make sure the order of the values is in the same order as the columns in the table.

INSERT INTO TABLE\_NAME VALUES (value1,value2,value3,...valueN);

***SQL UPDATE Query***

The SQL UPDATE Query is used for modifying the existing records in a table. Here, the WHERE clause is combined with UPDATE query to update selected rows; otherwise, all the rows would be affected.

***Syntax***:

The basic syntax of UPDATE query with a WHERE clause is as follows:

UPDATE table\_name SET column1 = value1, column2 = value2...., columnN = valueN WHERE [condition];

You can combine N number of conditions using AND or OR operators. Given below is a simple example of how the update query works:

Table: Employees

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPID** | **EMPName** | **EMPCity** | **EMPMobile** |
| 101 | Harish | New Delhi | 8912312390 |
| 102 | Janardan | Kanpur | 8812121212 |
| 102 | Janardan | Kanpur | 9900012222 |
| 103 | Ramnath | Chennai | 7778881212 |
| 104 | Laxman | Bangalore | 9990000123 |
| 104 | Laxman | Bangalore | 8123450987 |

#### **Update Employee**

SET column3 = value3, column4 = value4, ...

WHERE condition

SET = EMP City = ‘Pune’, EMP Mobile = '754498763'

WHERE EMPID = 101;

***SQL DELETE Query***

The SQL DELETE Query is used for deleting the existing records from a table. A WHERE clause with A DELETE query is used for deleting selected rows; otherwise, all the records would be deleted.

***Syntax***:

DELETE FROM Customers

WHERE Employee ID='101';

You can combine N number of conditions using AND or OR operators.

## **DQL - Data Query Language:**

***SQL SELECT Query***

SQL SELECT Statement is used for fetching the data from one or more database tables. The required data is returned in the form of A result table called the result-sets.

***Syntax***:

SELECT column1, column2, columnN

FROM table\_name;

Here, column1 and column2 are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

SELECT \* FROM table\_name;

***SQL Clauses ( WHERE, ORDER BY, GROUP BY, HAVING)***

***SQL WHERE***

The SQL WHERE clause is used for specifying a condition while fetching the data from a single table or joining with multiple tables.

If the given condition is satisfied, then it will return only specific values from the table. You would use a WHERE clause to filter the records, fetching only necessary records.

The WHERE clause is not only used in a SELECT statement, but also in an UPDATE and DELETE statement.

***Syntax***:

SELECT statement with WHERE clause:

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition]

***WHERE with Conjunctive Operators***

* The WHERE clause when combined with AND, OR, and NOT operators is used for filtering records based on more than one condition:
  + The AND operator displays a record if all the conditions separated by AND are TRUE.
  + The OR operator displays a record if any of the conditions separated by OR is TRUE.
  + The NOT operator displays a record if the condition(s) is NOT TRUE.

These operators also provide a means to make multiple comparisons with different operators in the same SQL statement.

***AND***

***Syntax***::

SELECT column1, column2, columnN …

FROM table\_name

WHERE [condition1] AND [condition2]...

AND [conditionN]…;

***NOT***

SELECT column1, column2, ...

FROM table\_name

WHERE NOT condition;

***QL ORDER BY***

The SQL ORDER BY clause is used for sorting the data in ascending or descending order, in one or more columns.

*Note: Some databases sort query results in ascending order by default.*

***Syntax***:

SELECT column-list

FROM table\_name

ORDER BY column1, column2, .. columnN ASC | DESC;

***SQL GROUP BY***

The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

* The GROUP BY clause groups records into summary rows
* GROUP BY returns one record for each group
* GROUP BY typically also involves aggregates: COUNT, MAX, SUM, AVG, etc.
* GROUP BY can group one or more columns

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

***Syntax***:

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

***SQL HAVING CLAUSE***

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

***Syntax***

SELECT column1, column2

FROM table1, table2

WHERE [ conditions ]

GROUP BY column1, column2

HAVING [ conditions ]

ORDER BY column1, column2

***SQL Wildcard Characters***

* A wildcard character is used for substituting any other character(s) in a string.
* Wildcard characters are used in conjunction with the SQL LIKE operator in the WHERE clause for searching a specified pattern in a column, e.g. for searching similar values etc. The following two wildcards are used in conjunction with the LIKE operator: The percent sign (%) and the underscore (\_).
* The percent sign (%) represents zero, one, or multiple characters. The underscore (\_) represents a single number or character. The symbols can be used in combination.

***Syntax***:

SELECT FROM table\_name WHERE column LIKE 'XXXX%'

|  |  |
| --- | --- |
| **Sr.No.** | **Statement & Description** |
| 1 | **WHERE SALARY LIKE '200%'**  Finds any values that start with 200. |
| 2 | **WHERE SALARY LIKE '%200%'**  Finds any values that has 200 in any position. |
| 3 | **WHERE SALARY LIKE '\_00%'**  Finds any values that has 00 in the second and third positions. |
| 4 | **WHERE SALARY LIKE '2\_%\_%'**  Finds any values that starts with 2 and are at least 3 characters in length. |
| 5 | **WHERE SALARY LIKE '%2'**  Finds any values that ends with 2. |

## **SQL Integrity Constraints**

Constraints are the rules enforced on data columns of a table, limiting the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

The constraints that are available in SQL are Primary Key, Foreign Key, Not Null, Unique, and Check. These constraints can be inserted with the CREATE TABLE Statement, or can also be specified in an already existing table by an ALTER TABLE statement.

Constraints can be defined in two ways:

1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table-level definition.

#### **NOT NULL Constraint:**

A column can hold NULL values (unknown data) by default. However, if you do not want a column to have a NULL value, such constraint needs to be defined for that particular column. For example, when the SQL creates a new table called CUSTOMERS and adds five columns (three of which are named ID, NAME, and AGE), they are specified not to accept NULLs:

#### **Syntax**

1. For A New Table

CREATE TABLE CUSTOMERS

( ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

1. For an Already Existing Table

If CUSTOMERS table is already there, a NOT NULL constraint to SALARY column in Oracle and MySQL is added as follows:

ALTER TABLE CUSTOMERS

MODIFY SALARY DECIMAL (18, 2) NOT NULL;

#### **UNIQUE Constraint:**

The function of a UNIQUE Constraint is to prevent two records having identical values in a particular column. In the CUSTOMERS table, for example, the constraint of two or more people having identical age in 2 or more records may be defined as follows:

***Syntax***

CREATE TABLE CUSTOMERS

( ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL UNIQUE,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

## **PRIMARY Key:**

The purpose of a primary key is to introduce a field in a table to uniquely identify each row/record in a database table.

* Primary keys must contain unique values.
* A primary key column cannot have NULL values.
* A table can have only one primary key that, when consisting of multiple fields, is known as a composite key.
* If a table has a primary key defined on any field(s), then two records having the same value of that field(s) cannot exist.

#### **Syntax**

a) For A New Table (Single Column)

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

b) For an Already Existing Table(Single Column)

ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);

NOTE: If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared not to contain NULL values (when the table was first created).

c) For an Already Existing Table(Multiple Columns)

ALTER TABLE CUSTOMERS

ADD CONSTRAINT PK\_CUSTID PRIMARY KEY (ID, NAME);

#### **Delete Primary Key:**

The Primary Key constraints from the table can be removed as follows:

ALTER TABLE CUSTOMERS DROP PRIMARY KEY ;

**FOREIGN Key:**

A Foreign Key or, a Referencing Key, is used for linking two tables together. When a Primary Key field from one table is inserted into the other table, it becomes a Foreign Key. So, a Foreign Key is a column or a combination of columns, whose values matches a Primary Key in a different table.

#### **Example: (Creating a Foreign Key)**

Consider the structure of the following two tables:

CUSTOMERS table:

CREATE TABLE CUSTOMERS (

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

If an ORDERS table is already there, the Foreign Key can be specified by the use of the following syntax:

#### **ALTER TABLE ORDERS**

ADD FOREIGN KEY (Customer\_ID) REFERENCES CUSTOMERS (ID);

#### **DROP a FOREIGN KEY Constraint:**

#### **Syntax**

ALTER TABLE ORDERS

DROP FOREIGN KEY;

#### **CHECK Constraint:**

The CHECK Constraint, when inserted in a column, ensures that all values that are entered satisfy certain conditions. If the condition is not found to be true, the record violates the constraint and the value is not entered into the table.

Create a new table called CUSTOMERS by having five columns.

Now add a CHECK constraint within the AGE column, so that it cannot have any CUSTOMER below 18 years old:

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL CHECK (AGE >= 18),

ADDRESS CHAR (25) ,

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

# SQL Other Topics

## ***SQL Joins***

The SQL Joins clause is used for combing or relating records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each (i.e. a customer who purchased a product and the quantity of that product).

## **SQL Join Types***:*

#### **INNER JOIN**

The most frequently used and important of the joins is the INNER JOIN. The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2). The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. After this, the column values for each matched pair of rows A and B are combined into a result row.

Syntax:

SELECT table1.column1, table2.column2...

FROM table1 INNER JOIN table2 ON table1.common\_filed = table2.common\_field;

***Example:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Order ID** | **CustomerID** | **OrderDate** |  | **CustomerID** | CustName | City |
| 10034 | 15 | 05/06/17 |  | 15 | N. Gopal | Pune |
| 10035 | 92 | 06/06/17 |  | 92 | B. Badri | Nagpur |
| 10036 | 40 | 06/06/17 |  | 40 | P.Raste | Delhi |

Now the relationship between two columns is the CustomerID, INNER JOIN will select columns that have the matching values in both records.

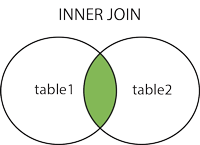
#### **Syntax**

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

The result of the INNER JOIN is represented in the table and the figure given below:

Table 3

|  |  |  |
| --- | --- | --- |
| ORDERID | CUSTNAME | ORDERDATE |
| 10034 | N. Gopal | 05/06/17 |
| 10035 | B. Badri | 06/06/17 |
| 10036 | P.Raste | 06/06/17 |

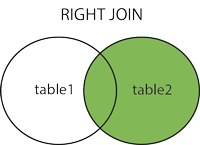
#### **Left Join**

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table, the JOIN will still return a row in the result. However, the row will have NULL in each column from the right table, plus matched values from the right table.

***Syntax***:

SELECT table1.column1, table2.column2... FROM table1 LEFT JOIN table2 ON table1.common\_filed = table2.common\_field.

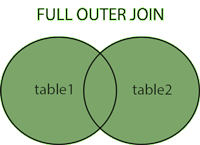
Here, a given condition could be any given expression based on your requirements.

***Right Join***

The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the JOIN will still return a row in the result. However, the row will have NULL in each column from left table, plus matched values from the left table.

***Syntax:***

SELECT table1.column1, table2.column2... FROM table1 RIGHT JOIN table2 ON table1.common\_filed = table2.common\_field;

***Full Join***

The SQL FULL JOIN combines the results of both left and right outer joins. The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

***Syntax:***

SELECT table1.column1, table2.column2... FROM table1 FULL JOIN table2 ON table1.common\_filed = table2.common\_field;

Here, the given condition could be any given expression based on your requirements.

***Self Join***

The SQL SELF JOIN is used for joining a table to itself as if the table were two tables. This action temporarily renames at least one table in the SQL statement.

***Syntax:***

SELECT a.column\_name, b.column\_name... FROM table1 a, table1 b WHERE a.common\_filed = b.common\_field;

Here, the WHERE clause could be any given expression based on your requirements.

***Cartesian Join***

If an SQL JOIN condition is omitted or invalid, the JOIN operation will result in a Cartesian Product: a number of rows equal to the product of all rows in the tables being joined.

***Syntax:***

SELECT table1.column1, table2.column2... FROM table1, table2 [, table3 ]

If the first table has 20 rows and the second table has 10 rows, the result will be 20 \* 10, or 200 rows. This query takes a long time to execute.

## **SQL Indexes**

Index in SQL is created on existing tables (on a single column or a group of columns) to retrieve the rows quickly when there are thousands of records. An index first sorts the data and then assigns a ROW ID for each row.

***SQL index notes***:

* An index helps to speed up SELECT queries and WHERE clauses.
* Since an index slows down DML operations, indexes should be used only on columns which are to be searched frequently.
* Indexes can be created or dropped with no effect on the data.

Creating an index involves the CREATE INDEX statement, which allows you to name the index, specify the table and column(s) to index, and indicate whether the index is in ascending or descending order.

Indexes can also be unique (similar to the UNIQUE constraint) in that the index prevents duplicate entries in a column or combination of columns on which there is an index.

#### **Syntax**

CREATE INDEX index\_name ON table\_name;

#### **Single-Column Indexes:**

CREATE INDEX index\_name ON table\_name (column\_name);

#### **Unique Indexes:**

Unique indexes are used not only for performance, but also for data integrity. A Unique index does not allow any duplicate values to be inserted into the table. CREATE UNIQUE INDEX index\_name on table\_name (column\_name);

**SQL - Using Views**

* A view is an SQL statement or predefined SQL query of the table stored in the database with an associated name. A view can contain all rows of a table or select rows from a table.
* A view can be created from one or many tables.
* Views share virtual tables which users find natural or intuitive. These provide restricted access to the data so that users can see and modify exactly what they need and nothing more.
* Views summarize data from various tables which can be used to generate reports.

***Syntax:***

CREATE VIEW view\_name AS SELECT column1, column2.....

FROM table\_name

WHERE [condition];

## **SQL Transactions**

Transactions are units or sequences of work performed in a logical order. Creating, updating, or deleting records are all transactions which need to be controlled for ensuring data integrity and avoid database errors.

These can be performed in a manual fashion by a user or automatically by some sort of a database program.

Practically, many SQL queries are plugged into a group and are executed together as part of a transaction.

All Transactions follow four standard properties referred to by the acronym ACID:

1. ***Atomicity***:

It ensures all operations within the work unit are completed successfully. If the transaction is aborted at the point of failure, all previous operations roll back to their former state.

1. ***Consistency***:

When a successfully committed transaction is performed, it ensures that the database changes properly and the data that has been modified complies with the constraints on the columns.

1. ***Isolation***:

It enables transactions to be transparent but operate independently of each other.

1. ***Durability***:

It ensures a committed transaction to persist in case of a system failure.

## **Transaction Control**

Transactional control commands can only be used with the DML (INSERT, UPDATE, DELETE). Such commands cannot be used while creating or dropping tables because these operations are automatically committed in the database.

## **The COMMIT and ROLLBACK Command:**

The command COMMIT saves all the invoked changes by transactions to the database, since the last COMMIT or ROLLBACK command. The ROLLBACK command can undo transactions.

#### **Syntax**

COMMIT

SQL> DELETE FROM CUSTOMERS

WHERE ADDRESS = DELHI;

SQL> COMMIT;

## **The SAVEPOINT Command:**

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. However, this command serves only in the creation of a SAVEPOINT among transactional statements. The ROLLBACK command is used for undoing a group of transactions.

#### **Syntax**

SQL> SAVEPOINT SP1;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=1;

1 row deleted.

SQL> SAVEPOINT SP2;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=2;

1 row deleted.

SQL> SAVEPOINT SP3;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=3;

1 row deleted

However, if for some reason rollback is decided till SP2, then the syntax for ROLLBACK will be as follows:

SQL> ROLLBACK TO SP2;

Rollback complete

So, the last two deletions will be rolled back.

***References***:

1. <https://www.w3schools.com/sql/>
2. <https://www.tutorialspoint.com/sql>
3. <https://beginner-sql-tutorial.com/>
4. <https://www.businessnewsdaily.com/5804-what-is-sql.html>

# Module 8

## UML Basics

## **Concepts of UML:**

The Object Management Group is an international (open membership) not-for-profit technology standards consortium that developed a unifying diagramming tool which was independent of a programming language. The tool was then named a UML (Unifying Modeling Language), which not only enabled IT professionals to model computer applications objectively, but also provided business users a way to obtain a better understanding of a system. A UML has the following characteristics:

* UML is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems.
* UML is very much different from the other programming languages like C++, Java, COBOL etc. because it is a pictorial language used for making software blueprints.
* Though the UML is not a programming language, its tools or UML diagrams can be used for generating code in various languages.
* UML has a direct relation with object-oriented analysis and design.
* Although UML is generally used in the modelling of software systems, it can also be used for modelling non-software systems like the process flow in a manufacturing unit etc.

#### **Goals of UML**

Computer Science, as you already know, is the study of algorithms and data. In object-oriented languages, algorithms are expressed by defining ‘objects’ which interact with each other. Object-oriented languages dominate the programming world because they model real-world objects. UML is a combination of several object-oriented notations: Object-Oriented Design, Object Modeling Technique, and Object-Oriented Software Engineering.

Using the strengths of the above three approaches, UML presents an easier, consistent methodology representing the best practices for building and documenting different aspects of software and business system modeling.

#### **UML 2.0 Specifications:**

UML 2.0 extends the original UML specifications to cover a range of software development efforts including agile practices. The number of specified diagrams in UML has been raised to 13 to include 4 new diagrams – communication, composite structure, interaction overview, and the timing diagram. The state chart diagrams have been renamed as a ‘State Machine Diagram”. UML 2.0 specification serves the following two functions:

* Improves integration between structural models of UML such as with class and activity diagrams.
* Defines a hierarchy of a software system by dividing it into components and subcomponents.

**A Conceptual Model of UML**

A conceptual model is the first step before drawing a UML diagram that describes concepts and their relationships in some domain. It follows the following rules:

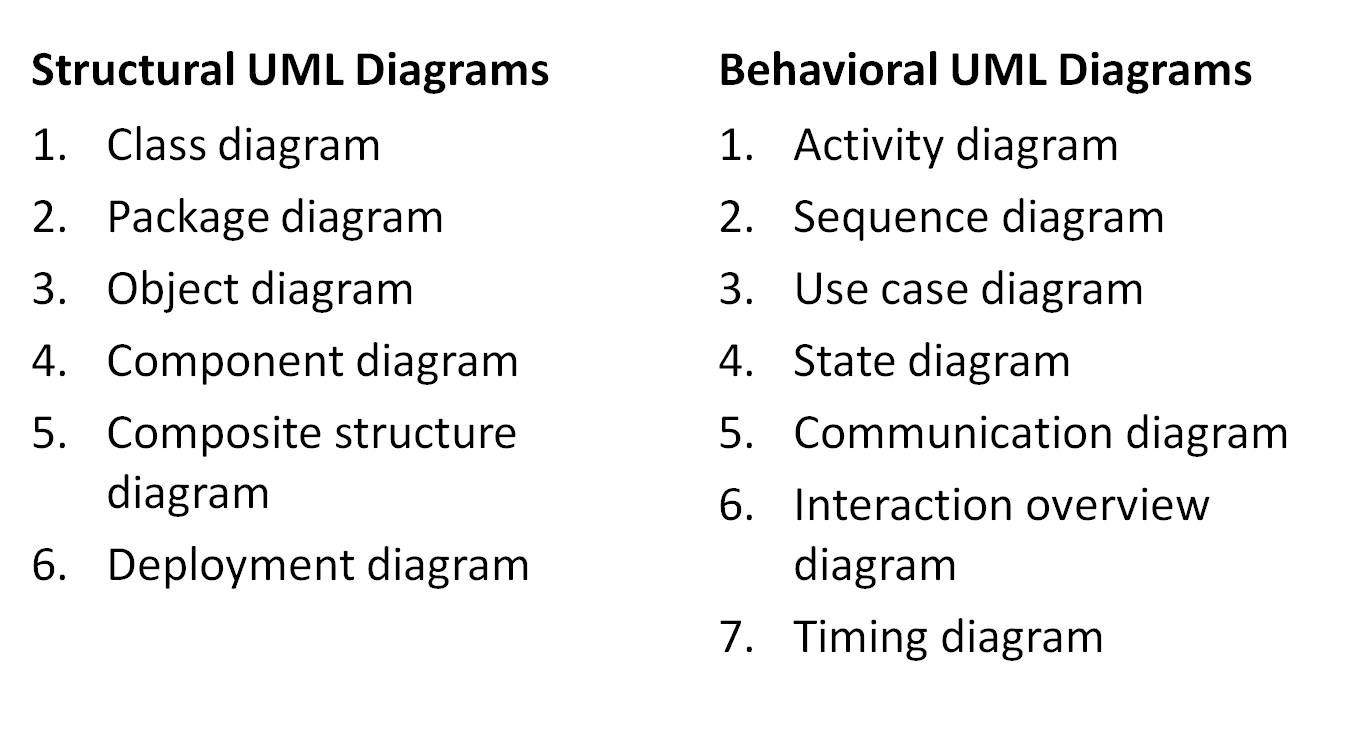
* Concepts are represented by Classes
* Relationships are represented by Lines
* Packages represent groups of related Classes and Associations
* Associations represent the functionalities of the system
* Objects represent specific instances of classes in real-time
  + Ex) A business might organize its’ concepts (classes) and relationships (Associations) into three packages

## **UML Diagrams**

The main objective of the UML diagrams is to define:

1. The structure of any system
2. The classes or objects of the system
3. The operations and attributes of the classes and objects
4. Relationships between different classes and objects

The UML diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams:



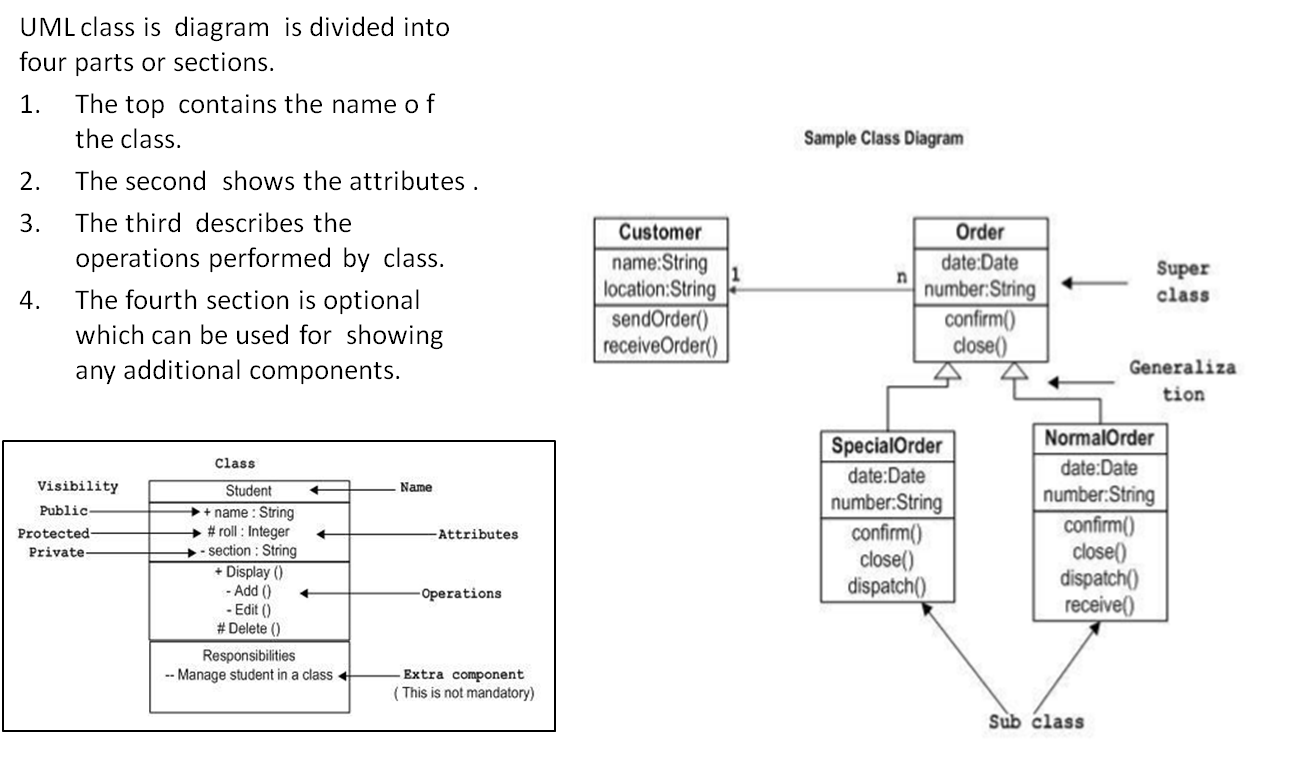
In the Behavioral UML Diagrams, the last 4 diagrams (State, Communication, Interaction Overview, and Timing) are true Interaction Diagrams.

## **Structural UML Diagrams:**

#### **Class Diagram**

A class diagram is the primary block for any object-oriented solution. This diagram provides a detailed model to translate into programming codes. It has the following primary goals:

* To represent the object-oriented view of a system in a static manner
  + Generally used for development



## **Package Diagram**Simple Package Diagram Example

As the name suggests, the objective of a package diagram is to show the dependencies between different packages in a system. Package diagrams are generally used for structuring high level system elements and to simplify complex class diagrams by grouping classes into packages. Packages are depicted as file folders and can be used for any of the UML diagrams.

The diagram to the right is a business model in which the classes are grouped into packages:

* Packages appear as rectangles with small tabs at the top
* The package name is on the tab or inside the rectangle
* The dotted arrows are dependencies

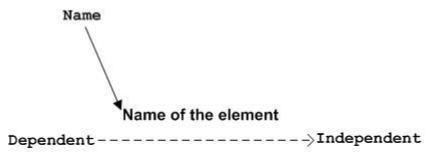
## **Object Diagram**

The object is represented in the same way as the class, therefore, it has the same usage as the class. An object is the actual implementation of a class, known as the ‘instance of a class’. An object diagram always uses real-world examples to show how a system will look at any given time or is used to build a prototype of a system from a practical perspective. Because there is data available in the objects, they are used to explain complex relationships between objects.

The object diagram serves the following functions:

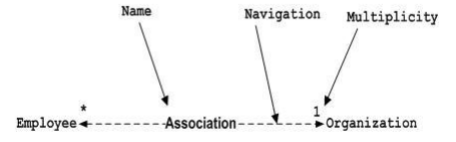
1. Provides a static view of an interaction
2. Helps in forward and reverse engineering
3. Defines the object relationships of a system
4. Understands object behavior and their relationship from practical perspective

***There are four kinds of relationships available***.

1. **Dependency**: is a relationship between two things in which change in one element also affects the other one

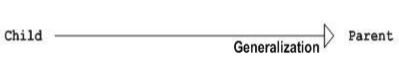
**The Dependency Notation**:

Dependency is represented by a dotted arrow, as shown in the diagram. The arrow head represents the independent element and the other end, the dependent element.

1. **Association**: Association is a set of links that connects elements of a UML model. It also describes how many objects or elements are taking part in that relationship. 

**Association Notation:**

Association is represented by a dotted line with or without arrows on both sides. The two ends represent two associated elements as shown in the diagram. The multiplicity is also mentioned at the ends (1, \* etc.) to show how many objects are associated.

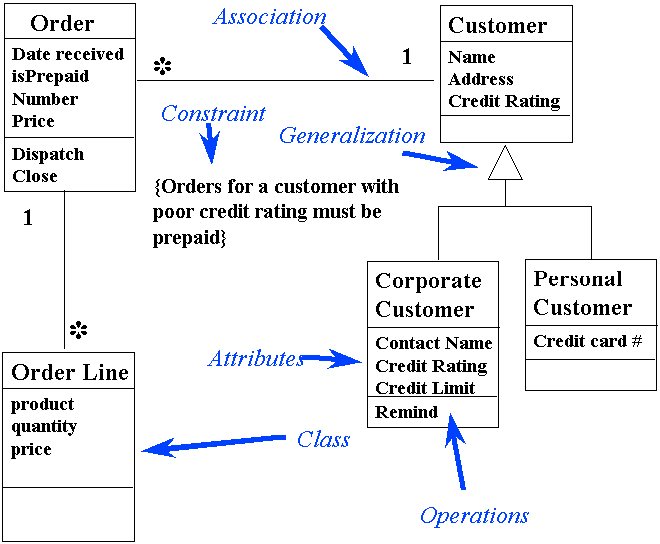
1. **Generalization**: The generalization relationship describes the inherited relationship of the object-oriented world and how a specialized element relates to a generalized element. It is, in layman’s terms, a parent and child relationship. 

**Generalization Notation**:

The generalization notion is represented by an arrow with a hollow arrowhead as shown in the picture One end of the arrow represents the parent element and the other end, the child element.

1. **Realization**: Realization is a specialized abstraction relationship between two sets of model elements. One element represents a specification or a responsibility later implemented by the other element.

**Realization Notation**Interface realization dependency from a classifier to an interface.

A realization dependency is shown as a dashed line with a triangular arrowhead at the end that corresponds to the realized element. 

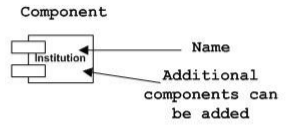
If interface is represented using the rectangle notation, interface realization dependency is denoted with interface realization arrow. The classifier at the tail of the arrow implements the interface at the head of the arrow.

**Object Diagram**

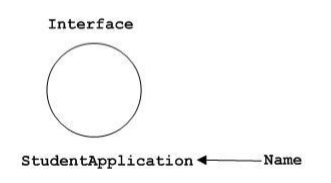
The diagram on the left is a class diagram, also seen as the object diagram. The diagram is depicting different types of relationships. The best way to illustrate what an object diagram looks like is to show the object diagram derived from the corresponding class diagram. An object diagram shows this relation between the instantiated classes and the defined class, and the relation between these objects in the system.

These object diagrams are useful in explaining smaller portions of a system, when a system class diagram is complex.

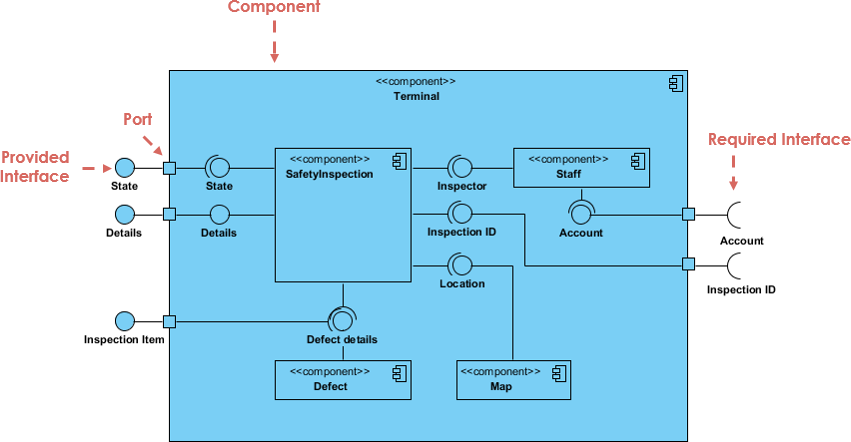
#### **Component Diagram**

This diagram describes the organization and the physical components in a system. In the world of UML 2.0, these components are conceptual stand-alone design elements (such as a business process) that provides or requires interfaces to interact with other constructs in the system. The physical elements described in UML 1.0, like files and documents, are now referred to as artifacts.

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner, with tabs or the word written above the name of the component to help distinguish it from a class.

***Interface Notation***

An interface (small circle or semicircle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input***.***



Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component. Dependencies among components are represented using dashed arrows.

## **Composite Structure Diagram**

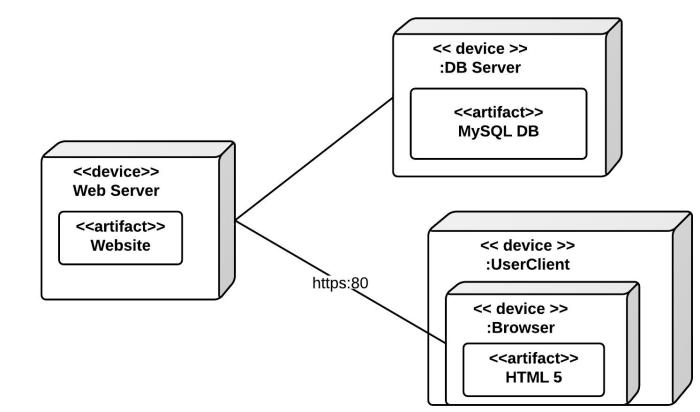
A Composite Structure Diagram has been one of the new artifacts added to UML 2.0. A composite structure diagram contains classes, interfaces, packages, and their relationships, to provide a logical view of the whole (or part of) a software system. It shows the internal structure (including parts and connectors) of a structured classifier or collaboration.

#### **Collaboration Notation**Image result for composite structure diagram in uml

Collaboration in a structure diagram is represented by a dotted eclipse, as shown below. It has a name written inside the eclipse. Collaboration represents responsibilities and, generally, responsibilities are in a group.

Though the role of a composite structure diagram is similar to that of a class diagram, it also describes in detail the internal structure and interactions among multiple classes. The inner classes and parts are represented graphically to show associations both between and within classes.

***Deployment Diagram***

A deployment diagram in UML describes an aspect of the system itself: the physical deployment of information generated by the software program on hardware components. The information that is generated by the software is called an artifact. 

Deployment diagrams are made up of several UML shapes. Nodes, the three-dimensional boxes, represent the basic software or hardware elements. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

Deployment Diagrams are used for the following:

* To show which software elements are deployed by which hardware elements
* To illustrate the runtime processing for hardware
* To provide a view of the hardware system’s topology
* The different parts of the web server and how Javascript works on the User Client can also be included in this diagram
  + The example on your rights shows a basic deployment diagram. There is a web server, a database server, and the user's machine where they view the website.

**Courtesy**: <https://www.lucidchart.com/pages/uml/deployment-diagram>

## **UML Behavioral Diagrams:**

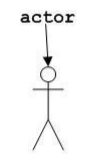
#### **Activity Diagrams**Image result for uml activity diagram

An activity diagram presents a series of actions or the flow of control in a system. It consists of activities and links that can also describe the steps in a use-case diagram. The activities that are depicted can be sequential, concurrent, or branched. An activity diagram will have a beginning (a start point) marked by a small filled circle followed by an arrow. An action state is represented by a rectangle with rounded corners, and an action flow is represented by simply arrows.

## **Sequence Diagrams:**

A sequence UML diagram represents the interaction of different objects in a system, the way they operate with one another, and the order of their interaction. To say it simply, a sequence diagram shows object interactions arranged by time sequence that serve the following objectives:

* Represent the details of a UML use case and understands the system functionality
* Provides a model for understanding the logic of a sophisticated procedure, function, or operation and views how tasks are moved between objects or components of a process

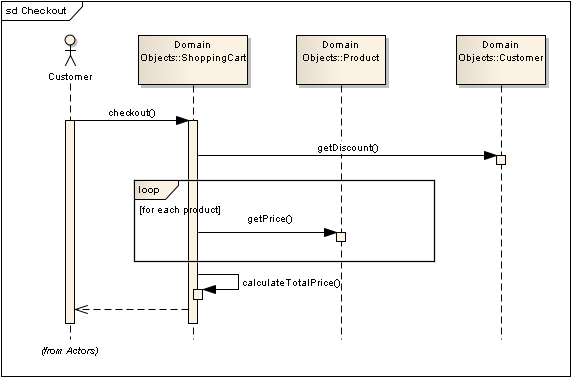
**Actors**: Actors are entities that, although external, interactive with the system 

**Object Symbol**: The solid rectangle represents a class, or object, to show how an object will behave in the context of the system. It cannot list attributes.

**Activity Box**: The diagram on the right represents the time needed for an object to complete a task. The longer the task, the longer the activation box will be.

**Lifeline**: A dashed vertical line (shown below) represents the passage of time and the sequence of an event as it extends downwards. The lifeline begins with a labelled rectangle shape or an actor symbol.

#### **Sequence Diagram for Online Purchase*:***



The above sequence diagram represents the online purchase by a customer and the messages flowing from one object to another. The notations and shapes that are represented in the above diagram have the following meaning to convey:

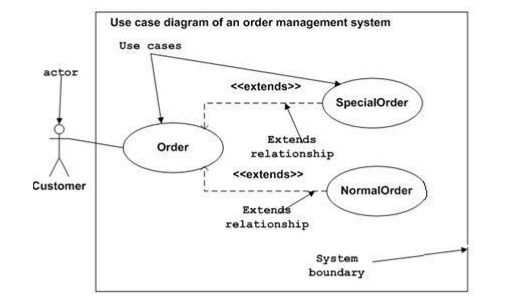
**Option Loop Symbol**: A rectangle shape with a smaller label that models "if then" scenarios, i.e., a circumstance that will only occur under certain conditions.

**Synchronous Message Symbol**: A solid line with a solid arrowhead used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.

**Return Message Symbol**: A dashed line with a lined arrowhead.

## **Use Case Diagram:**

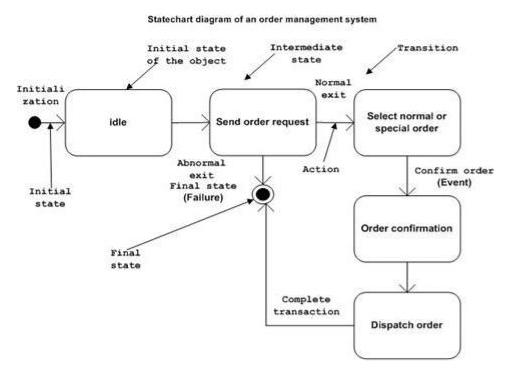
Use case diagrams are the diagrams that describe a set of actions (use cases) which some system(s) (subject) should or can perform in collaboration with one or more external users of the system (actors).

A use case diagram is represented as an eclipse with a name inside it. It may contain additional responsibilities. The use case is used for capturing high level functionalities of a system and their internal/external controllers. These controllers are known as actors. Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Function of use case diagrams are in fact twofold – they are

Behavior diagrams and structure diagrams are special cases of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations. This is because they describe the behavior of the system.

#### **State Diagram:**

Any real-time system is expected to be reacted by some kind of internal/external events. The state diagram depicts the behavior of the system over a time frame. It basically describes the state change of a class, interface etc. This diagram is typically drawn for every class in the system that contains significant dynamic behavior. 

The behavior of the class is modeled in terms of the state it is in at various times, what actions it performs in various states, and also when it is in transition—the class is going from one state to another state based on events in the system.

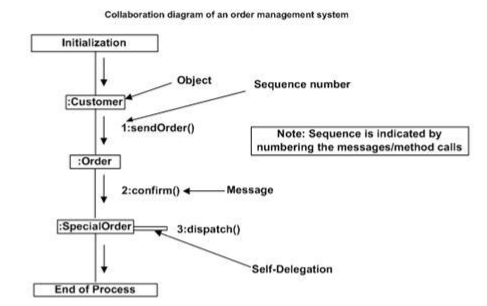
States represent the conditions of objects at certain points in time. Events represent incidents that cause objects to change from one state to another, depicted through transition lines that represent movement from one state to another. Actions occur when an object arrives in a state.

*Note: The UML state diagram is based on the work of David Harel, and is also referred to as a Harel State chart Diagram.*

## **Communication Diagram**

The Communication diagram is an interaction diagram that was introduced in UML 2.0, earlier known as a collaboration diagram.

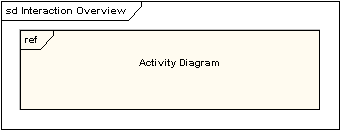
1. It provides similar information to a sequence diagram but its focus is on the relation of objects.
2. A communication diagram can also be called as an extension of an object diagram.
3. Objects are shown with association connectors between them.
4. Messages are shown as short arrows pointing in the direction of the message flow.
5. The sequence of messages is shown through a numbering scheme.
6. Return values may also be included/numbered to indicate when they are returned within the time sequence.



**Table: The difference between Sequence Diagram and Communication Diagram**

|  |  |
| --- | --- |
| **Sequence Diagram** | **Communication Diagram** |
| Explicitly shows the sequence of messages. | Shows relations (links) in addition to interactions. |
| Shows execution occurrence. | Provides an understanding of the pattern of communication. |
| Better for visualizing overall flow. | Provides a better visualization of the effects on an object. |
| Better for complex or real-time situations. | Good for brainstorming sessions. |
| Can be used for translating the use case scenario into a visual tool for system analysis. | Provides a graphic format and thus, is good for modeling the effects on objects. |

## **Interaction Overview Diagram**

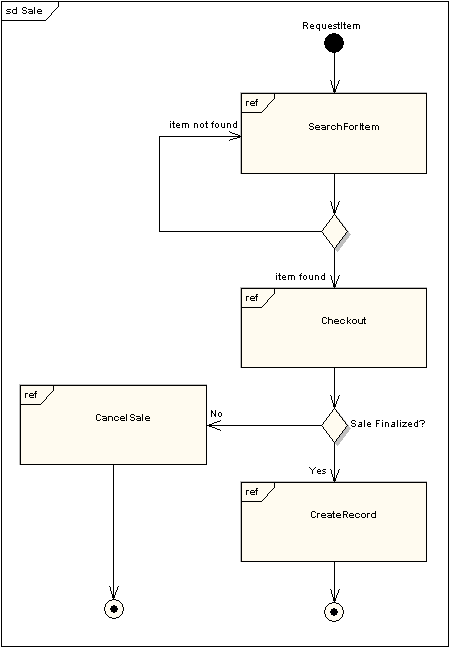
An Interaction overview diagram can be called a ‘combination of activity and sequence’ diagram. Most of the notations for interaction overview diagrams (initial, final, decision, merge, fork and join nodes) are the same as the activity diagrams. However, these diagrams also include two new elements: interaction occurrences and interaction elements. 

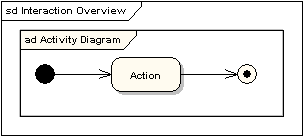
#### **Interaction Occurrences:**

These provide references to the existing interaction diagrams. An interaction occurrence is shown as a reference frame which has "ref" written in the top-left corner. The name of the diagram being referenced is shown in the center of the frame.

## **Interaction Overview Diagram for Sale Process:**

#### **Interaction Elements**

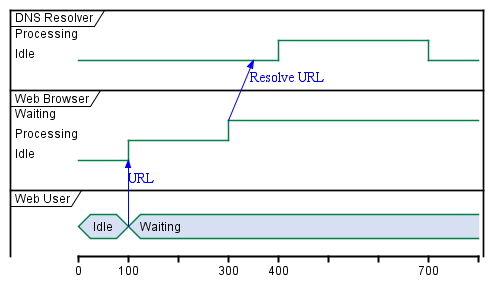
Interaction elements also display a representation of existing interaction diagrams within a rectangular frame. However, they differ from interaction occurrences in that they display the contents of the references diagram inline.



The example on the right depicts a sample sale process. Here, the sub-processes have been abstracted within interaction occurrences.

#### **Timing Diagrams**

A timing diagram is a specific behavioral UML diagram that focuses on timing constraints. The time diagram describes the behavior of how objects interact with each other during a certain period of time. A timing diagram can also be thought of as an inverted sequence diagram. On the x-axis, instead of different objects, time passes from left to right. The different components of the system that interact are stacked together on the y-axis.



**References:**

1. <https://www.smartdraw.com/uml-diagram/#UMLMeaning>
2. <https://www.tutorialspoint.com/uml/uml_tutorial.pdf>
3. <https://www.smartdraw.com/component-diagram>
4. <https://creately.com/blog/diagrams/uml-diagram-types-examples/>
5. <https://www.lucidchart.com/pages/uml-sequence-diagram>
6. <https://www.scribd.com/document/93308585/UNIX-Tutorials>
7. [http://www.sunnyoasis.com/services/UNIXhelp.html](http://www.sunnyoasis.com/services/unixhelp.html)

# Module 9

## **UNIX Fundamentals**

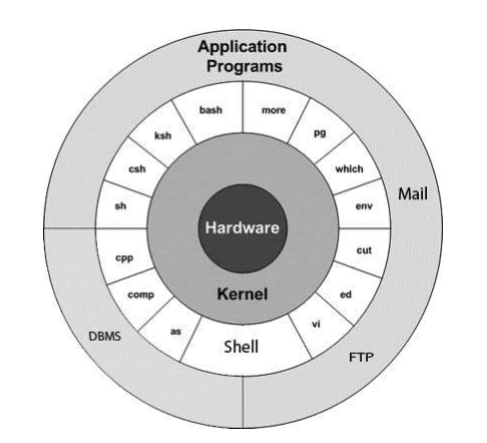
## **The UNIX Operating System**

An operating system is a set of programs that acts as a link between the computer and the user. The job of an operating system is to allocate the system resources and coordinate with computer's internals.

#### **Special Features of the UNIX Operating System**

* UNIX is a multi-user, multi-tasking operating system.
* PC operating systems such as MS-DOS or MS-Window only allow multitasking (different programs to run on the same OS simultaneously), whereas UNIX allows multiple users to work together on the same operating system.
* UNIX was one of the first operating systems to be written in C language.
* UNIX is an operating system that is extensively used by engineering and scientific workstations.
* UNIX is a machine independent operating system which means that it can be used by a variety of computer hardware.
* UNIX treats all types of files as simple byte arrays, which is a simplified ‘file model’ as compared to other operating systems.
* UNIX is the first open or standard operating system that can be improved or enhanced by anyone.
* UNIX has been developed in a way to facilitate a software development environment.

#### **Variants of UNIX**

The UNIX operating system was originally developed in 1969 at Bell Labs by a group of AT&T employees, Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna. There are various UNIX variants available in the market. 

Solaris UNIX, AIX, HP UNIX, and Linux are a few examples that are freely available.

## **UNIX Architecture**

UNIX has the following four basic components:

1. Kernel
2. Shell
3. Commands and Utilities
4. Files and Directories

## **Kernel**

The kernel is the heart of the operating system. It acts as the memory-resident control program loaded at system start up (boot).

The kernel performs the following functions:

1. Interprets and executes instructions from the shell
2. Interacts with the hardware
3. Memory management
4. Inter-process communication
5. I/O device management
6. Task scheduling and file management

#### **Shell**

The function of the shell or the command line interpreter is to translate/convert the commands entered by the user into a language that is understood by the Kernel.

The shell uses standard syntax for all commands. Each shell has its own programming language. It uses "shell scripts” or command files to accomplish a series of tasks.

***Commands and Utilities***

UNIX provides several hundred utility programs, or commands, for the following functionalities:

* Editing
* File maintenance
* Printing
* Sorting
* Programming support
* Online info

## **Files and Directories**

All data in UNIX is organized into files and all files are organized into directories. These directories are organized into a tree-like structure called the file system.

#### **Getting Started with UNIX**

#### **Starting the System**

#### **System Boot up**

The boot sequence starts when the computer is turned on, and is completed when the kernel is initialized and the system is launched. Finally, it prompts the user to log into the system, after which they can use it for day-to-day activities.

#### **Login UNIX**

When you first connect to a UNIX system, there is usually a login prompt similar to the following:

login:

#### **To log in:**

1. Keep your user-id (user identification) and password ready. Contact your system administrator if you do not have this information yet.
2. Type your user-id at the login prompt. A user-id is always case-sensitive, so be sure to type it exactly as the instructions of the administrator.
3. Type your password (it is also case sensitive) at the password prompt. Then, press ENTER.
4. After the correct user-id and password have been entered, the user will be allowed to enter into the system.

***Change Password***

All UNIX systems require passwords in order to ensure that one’s files and data remain intact and the system itself is secure from hackers and crackers. The password is changeable and known only to the user, not to the system staff. Here are the steps to change your password:

1. To start, type password at the command prompt
2. Enter your current password.
3. Type in your new password. Keep in mind that the password should be complex enough so that nobody can guess it, but make sure you remember it or can write it down.
4. The next step is to verify the password by typing it again.

***The UNIX Shell***

The shell provides an interface to the UNIX system. Since the shell wraps around the delicate interior of the operating system, it is called a shell.

The shell gathers input from the user and executes programs based on the input. When a program finishes executing, it displays the program's output. A shell is an environment in which a user can run commands, programs, and shell scripts.

#### **Shell Prompt**

The prompt ‘$,’ referred to as a command prompt, is issued by the shell. While the prompt is displayed, you can type a command.

***4.1 Types of Shells***

In UNIX there are two major types of shells:

1. The Bourne shell
   1. The default prompt for this shell is the $ character. It has the following derivatives:

POSIX shell (sh)

Korn Shell (Ksh)

Borne Again Shell (Bash) the most popular shell.

1. The C shell
   1. The prompt for this shell is %. It has the following derivatives:

C shell (csh)

TENEX /TOPS C shell (tcsh)

#### **Shell Scripts**

The shell script is a language that allows the automation of most command line tasks at the UNIX Shell. It is a list of commands listed in the order of execution. A good shell script will have comments, preceded by a pound sign #, that describe steps. Shell scripts and functions are interpreted but not compiled. A shell script consists of the following:

1. Conditional tests (whether value A is greater than value B)
2. Loops which allow users to go through massive amounts of data
3. Files and variables to read and store
4. The script may include functions

#### **The UNIX File System**

The peculiar thing about UNIX is that everything in UNIX is considered to be a file, including physical devices such as DVD-ROMs, USB devices, floppy drives, etc.

A file system is a logical collection of files on a partition or disk. A partition can be looked upon as a container for information that can occupy an entire hard drive if desired. The hard drive can have various partitions and each partition usually contains only one file system. One file system per partition allows for the logical maintenance and management of different file systems.

#### **UNIX File Management**

All data in UNIX is organized into files and all files are organized into directories. There are three basic types of files in UNIX:

1. **Ordinary Files**

An ordinary file is a file in the system that contains data, text, or program instructions.

2. **Directories**:

Directories store both special and ordinary files.

3. **Special Files**:

There are some special files that provide access to the hardware. A few examples of these include hard drives, CD- ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts that enable you to access a single file using different names.

#### **Some Basic UNIX Commands**

$ passwd – For changing password

$ls – To list all of the files or directories available in a directory

$ vi filename – To create a new file

$ cat filename – To display the contents of a file

$ cp source\_file destination\_file – To copy a file

$ rm filename – To delete a file

Meta Characters: (Characters that have special meaning in UNIX)

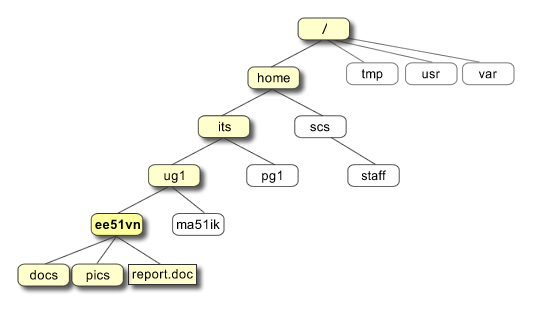
\* to match 0 or more characters

? matches with a single character

#### **UNIX Directories**

In UNIX, a directory is a file that performs the function of storing file names and related information. All types of files - ordinary, special, or directory - are contained in directories.

#### **The Root Directory**

A hierarchical structure, referred to as a directory tree, is followed for organizing files and directories. The tree has a single root node, the slash character ( /), which is known as the root directory. It is at the top and all other directories are contained below. In the diagram given below, one can see the home directory of a student "ee51vn" contains two sub-directories (docs and pics) and a file called report.doc.

* $cd ~ - Takes you to your home directory
* $pwd – Prints the current working directory
* $mkdir dirname – Makes a new directory
* $rmdir dirname – Removes a directory
* $cd dirname – Makes changes to any directory

## **Home Directory**

Each user has a unique \\\"home\\\" directory, that is part of the file system reserved for your files. After login, you are \\\"taken\\\" to your home directory automatically. This is where you start your work. You are in control of the home directory and files located there. You are also in control of the file access permissions (discussed below) to the files in your home directory. Generally, you alone should be in charge to create/delete/modify files in your home directory. Others may have permission to read or execute your files as you determine.

In most UNIX systems, you can \\\"move around\\\" or navigate other parts of the file system outside of your home directory. This depends upon how the file permissions have been set by others and/or the System Administrator.

## **Directory Structure**

As highlighted, UNIX uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A UNIX file system is a collection of files and directories that has the following properties:

1. It has a root directory (/) that contains other files and directories
2. Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an inode
3. By convention, the root directory has an inode number of 2 and the lost+found directory has an inode number of 3
4. Inode numbers 0 and 1 are not used. File inode numbers can be seen by specifying the -i option to ls command
5. It is self-contained; there are no dependencies between one file systems

#### **Some Special Directories**

|  |  |
| --- | --- |
| **Directory** | **Description** |
| **/** | The root directory that should contain only the directory needed at the top level of the file structure. |
| **/bin** | Contains the executable files available to all users. |
| **/dev** | Device drivers. |
| **/etc** | Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, hosts, where to send critical messages. |
| **/lib** | Contains shared library files and sometimes other kernel-related files. |
| **/boot** | Contains files for booting the system. |
| **/home** | Contains the home directory for users and other accounts. |
| **/proc** | Contains all processes marked as a file by process number along with other information that is dynamic to the system. |
| **/tmp** | Holds temporary files used between system boots. |
| **/usr** | Used for miscellaneous purposes, or can be used by many users. Includes administrative commands, shared files, library files, etc |
| **/var** | This location contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data. |
| **/sbin** | Contains binary (executable) files, usually for system administration. |
| **/kernel** | Contains kernel files |
| **/mnt** | This location is used for mounting other temporary file systems, such as a cdrom and floppy. |

## **UNIX File Permission Setup**

#### **Types of Permissions**

UNIX, being a multi-user system, uses permissions and ownership for security.

There are three types of users: Owner, Group and Other. There are also three types of file permissions – Read, Write and Execute (denoted by r, w, and x).

These permissions have the following attributes:

* **Owner permissions**: Determines what actions the owner of the file can perform on the file.
* **Group permissions**: Determines what actions a user, who is a member of the group that a file belongs to, can perform on the file.
* **Other (world) permissions**: Indicates what actions all other users can perform on the file.

#### **The Permission Indicators**

The ls -l command displays various information related to file permissions as follows:

$ls -l /home/amrood -rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir

Here, the first column represents different access modes i.e. permission associated with a file or directory.

The first '-' implies that we have selected a file.p>. Otherwise, if it were a directory, d would have been shown.

The permissions are broken into groups of threes and each position in the group denotes a specific permission in the order: read (r), write (w), execute (x).

The first three characters (2-4) represent the permissions for the file's owner. The first part of the code is 'rwx'. This suggests that the owner 'Home' can:

1. Read the file

2. Write or edit the file

3. Execute the file

The second group of three characters (5-7) consists of the permissions for a file that belongs to the whole group. For example, r-x represents that the group has read (r) and execute (x) permission, but no write permission is given since the write bit is set to '-'.

The last group of three characters (8-10) represents the permissions for everyone else. For example, r represents the ‘other world’ whereby any user has read (r) only permission. The permissions and ownership can also be changed.

***Changing the Permissions***

The permissions on a file can be changed by 'chmod' command syntax that consists of at least three parts from the following lists:

**Access Class Operator Access Type**

u (user) + (add access) r (read)

g (group) - (remove access) w (write)

o (other) = (set exact access) x (execute)

a (all: u, g, and o)

1. To add permissions for everyone to read a file in the current directory named myfile:

#### **Syntax**

chmod a+r myfile OR

chmod +r myfile (if the access class is not given, it's assumed to be all)

2. To remove, read, and write permissions for the group and other users in the directory named myfile:

#### **Syntax**

chmod go-rw myfile

#### **Changing the Ownership**

The 'chown' command can change the ownership of a file/directory.

1. Change the owner of the file.

#### **Syntax**

$ chown user1 sample.txt

2. Change the group of the file.

#### **Syntax**

$ chown :mygroup file.txt

3. Change both the owner and group of the file in a single command.

#### **Syntax**

$ chown user1:mygroup file.txt

4. The 'chgrp' command can change the group ownership

#### **Syntax**

chrgrp group filename

(group specifies the new group, and the file name or directory)

## **UNIX Environment**

The environment variables are used for defining the UNIX environment. Some variables are set by the system, by the users, by the shell, or any program that loads another program.

Environment variables are set without using $ sign. However, while accessing them, the $ sign is used as a prefix. These variables retain their values until the user comes out of the shell.

When the user logs in to the system, the shell undergoes a phase called initialization to set up various environments. This is usually a two-step process that involves the shell reading the following files:

• /etc/profile

• profile

The process is as follows:

1. First, the Shell checks to see whether the file /etc/profile exists.
2. If it exists, the shell reads it. Otherwise, this file is skipped. No error message is displayed.
3. The shell checks to see whether the file .profile exists in your home directory.
4. If it exists, the shell reads it; otherwise, the shell skips it. No error message is displayed.
5. As soon as both files have been read, the shell displays a prompt: $
6. This is the prompt where a user can enter the commands in order to execute them.

#### **The .profile File:**

The file /etc/profile is maintained by the system administrator of a user’s UNIX machine and contains the shell initialization information required by all users on a system. The file .profile is under a user’s control. A user can add any required shell customization information to a file. The minimum set of information that is needed to configure includes:

* The type of terminal to be used
* A list of directories in which commands are located
* A list of variables affecting the look and feel of a user terminal

As a user, you can check your .profile available in your home directory. Open it using the vi editor and check all the variables set for your environment.

#### **Setting the PATH:**

When you type any command on a command prompt, the shell has to locate the command before it can be executed. The PATH variable specifies the locations in which the shell should look for commands:

$PATH=/bin:/usr/bin

## **UNIX Communication**

Working in a distributed environment requires you to communicate with remote users and have access to remote UNIX machines. There are several UNIX utilities that are especially useful for users computing in a networked, distributed environment.

#### **The ping Utility:**

The ping command is used for sending an echo request to a host available on the network. Using this command, you can check if your remote host is responding well or not.

The ping command is also useful in the following situations:

• Tracking and isolating hardware and software problems

• Determining the status of the network and various foreign hosts

• Testing, measuring, and managing networks

#### **Syntax**

$ping hostname or ip-address

#### **The ftp Utility:**

Using this utility, a user can upload his/her file to another computer and download a file sent from another computer.

The ftp utility has its own set of UNIX like commands that allow you to perform tasks such as:

* Connect and login to a remote host
* Navigate directories
* List directory contents
* Put and get files
* Transfer files as ascii, ebcdic or binary

#### **Syntax:**

$ftp hostname or ip-address

#### **The telnet Utility:**

Often, a user requires connecting to a remote UNIX machine to work on it. This is facilitated by the Telnet utility.

#### **UNIX – The vi Editor**

The vi is generally considered the ‘de-facto’ standard in UNIX editors for the following reasons:

* 1. Availability on all flavors of the UNIX system
  2. Similar implementations across the board
  3. Minimal resource requirement
  4. Better user experience when compared to other editors like ed. The vi editor can edit an existing file, create a new file from scratch or can be used only to read a text file

$vi filename – Creates a new file if it does not already exist, otherwise it opens an existing file.

$vi -R filename – Opens an existing file in read only mode.

$view filename – Opens an existing file in read only mode.

## **Operation Modes:**

While working with the vi editor, you could come across the following two modes:

1. **Command Mode**:

In this mode, whatever a user types is interpreted as a command. This mode enables a user to perform administrative tasks such as saving files, executing commands, moving the cursor, cutting (yanking) and pasting lines or words, and finding and replacing.

1. **Insert Mode**:

This mode enables a user to insert text into the file. Everything that's typed in this mode is interpreted as input and finally it is put into the file. The vi always starts in command mode. However, to enter text, you must be in the insert mode. You can come in the insert mode by simply typing i.

To get out of insert mode, press the Esc key. This will put you back into the command mode.

*Hint: If you are not sure which mode you are in, press the Esc key twice, and you will be in command mode. Open a file using the vi editor and then start to type some characters. Next, enter into command mode to understand the difference.*

***Getting Out of vi:***

The command to quit out of vi is :q.

1. Once in command mode, type a colon, and 'q', followed by return.
2. If your file has been modified in any way, the editor will warn you of this and will not let you quit.
3. If you do not want to save changes, ignore the message, and type the command :q!. This lets you exit vi without saving any of the changes.
4. If you want to save changes, press w to save the contents of the editor.
5. You can combine the above command with the quit command :wq and return.
6. The easiest way to save your changes and exit out of vi is the ZZ command. In command mode, type ZZ, which is the equivalent of :wq.
7. You can specify a different file name to save to by specifying the name after the :w. For example, if you wanted to save the file you were working in as another filename, (ex. filename2), you would type :w filename2 and return.

## **User Administration**

#### **Types of Accounts**

Three types of accounts exist on a UNIX System:

* **Root account:** The root account holder is also called a super user. The super user has a complete and unrestricted control of the system and is allowed to run any commands. This user is, in fact, the system administrator.
* **System accounts**: System accounts are those needed for the operation of system-specific components. For example, mail accounts and the sshd accounts.
* **User accounts**: General users and groups of users get interactive access to the system through user accounts, who have limited access to critical system files and directories.

UNIX supports the concept of a Group Account that logically groups a number of accounts. Every account would be a part of any group account. The UNIX groups plays an important role in handling file permissions and process management.

## **Managing Users and Groups:**

The following are the main user administration files:

1. **/etc/password**: Keeps user account and password information. This file holds the majority of information about accounts on the UNIX system.
2. **/etc/shadow**: Holds the encrypted password of the corresponding account; not all systems support this file.
3. **/etc/group**: This file contains the group information for each account.
4. **/etc/gshadow**: This file contains secure group account information.

#### **User Administration Commands:**

|  |  |
| --- | --- |
| **Command** | **Description** |
| useradd | Adds accounts to the system. |
| Usermod | Modifies account attributes. |
| userdel | Deletes accounts from the system. |
| groupadd | Adds groups to the system. |
| Groupmod | Modifies group attributes |
| groupdel | Removes groups from the system. |

***References***:

1. [https://www.tutorialspoint.com/UNIX/UNIX-getting-started.htm](https://www.tutorialspoint.com/unix/unix-getting-started.htm)

2. [http://www.ee.surrey.ac.uk/Teaching/UNIX/UNIX1.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html)

3. <https://www.guru99.com/file-permissions.html>

4. <https://www.scribd.com/document/93308585/UNIX-Tutorials>

5. [http://www.sunnyoasis.com/services/UNIXhelp.html](http://www.sunnyoasis.com/services/unixhelp.html)

# Module 10

## Software Testing

Software testing is a verification activity to see whether the actual results match the expected results and the process to develop a defect-free software system. According to ANSI/IEEE 1059 standard, testing can be defined as “a process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item” (1. Quality Assurance and Quality Control).

It is important here to know here that testing is different from quality assurance (QA) and quality control (QC). Though closely related, these are different concepts which every software personnel needs to understand.

## **Quality Assurance**

QA is a set of pre-designed activities undertaken to ensure that the development and/or maintenance process meets all of its objectives.

* A set of activities that ensures the whole process of software development is defined, documented, and appropriate.
* Examples of QA activities are methodology and standards development for writing the source code.

#### **Quality Control**

Quality control (QC) is a procedure or set of procedures intended to ensure that a manufactured product or performed service adheres to a defined set of quality criteria or meets the requirements of the client or customer. QC is similar to, but not identical with, quality assurance (QA).

* A set of activities designed to evaluate a developed work product.
* A set of activities that focus on finding defects in a specific deliverable product.
* Activities that ensure the verification of a developed software with respect to the documented requirements.

Testing is one example of a QC activity. Both QA and QC activities are generally required for successful software development.

#### **Testing and Debugging**

**Testing**

* Testing is a process of executing a system with the intent of finding defects.
* Testing involves identifying bug/error/defect in software without correcting it.
* Professionals with a quality assurance background are involved in bug identification.

**Debugging**

* A process of identifying, isolating, and fixing the problems/bugs in the code.
* Debugging is the routine process which comprises locating and removing the software program abnormalities.
* The process of debugging is methodically handled by software programmers using debugging tools.
* Debugging is performed to allow the right program operations to conform to the given specifications and requirements.
* For large software programs that contain a million lines of source code, debugging is first completed for different software components separately and then for the program as a whole.

#### **Who does the Testing?**

Who performs the testing depends on the overall process of software development, project management, and the associated stakeholders of the project(s). In most large IT companies there is a ‘Testing Team’ specifically assigned the responsibility of evaluating the developed software/product, in the context of the given requirements. This approach to software project management is in accordance with the “waterfall model” (see Chapter 4: SDLC) where there is a separation between coders and testers.

The advantage of separating the duties of development and testing is that it involves more people (i.e. more opportunities to find bugs). In case some ambiguity exists in the requirements or specifications, it can be interpreted by the testers in different ways so as to obtain greater clarity.

Initial testing is also performed by developers, often referred to as Unit Testing. Reviewing the design in the design phase with an intent to improve the design is also considered testing. In general, the following software professionals are involved in testing within their respective responsibilities and capacities (also refer to Chapter 4: SDLC):

* Software Tester
* Software Developer
* Project Lead/Manager
* End User

IT companies assign the following professional designations for people involved in the testing of software:

* Software Tester
* Software Quality Assurance Engineer
* QA Analyst, etc.

#### **When to Start Testing?**

Testing as such is not a stand-alone activity. When and how testing is to be done depends upon the development model chosen for the project.

Most contemporary Software Development Life Cycle (SDLC) norms recommend testing to be started as early as the ‘Requirements Gathering Phase’ and continued until the deployment of the software. Some even recommend it until maintenance. In the V-model, there is a corresponding testing phase for each and every software development phase.

#### **Advantages of Timely Testing**

* Starting the testing process early not only reduces the cost, but also reduces time by avoiding rework. At the same time, it assists in producing error-free software delivered to the client on time.
* Like SDLC, there is a corresponding STLC (Software Testing Life Cycle) that consists of a series of methodological activities carried out by a team of testers for testing of a software product. In some instances, the term “testing” involves not just testers, but developers as well.

#### **Documentation**

Proper documentation is essential for successful testing. Testing documentation involves the documentation of all of the different artifacts that should be developed before or during the testing of a software.

(The artifact is one of many kinds of tangible byproducts produced during the development of a software product)

For effective QA and QC activities, documentation for the following is required:

1. How much testing effort is required?
2. What is the test coverage?
3. Defining the Resources
4. Test environment
5. Preparing the Test Scenarios
6. Defining the limitations of the testing
7. Test Schedule
8. Requirement of tracking/tracing, etc.

The following are the most commonly used documented artifacts:

* Test Plan
* Test Scenario
* Test Cases
* Traceability Matrix

#### **Test Plan**

A Test Plan includes the following:

* Introduction to the ‘Test Plan Document’
* Assumptions while testing the application
* Test scenarios
* List of test cases
* List of features to be tested
* The approach for testing the software
* List of deliverables
* The list of resources that will be allocated for testing
* Any risks involved during the testing process
* A schedule for tasks to be completed and milestones to be achieved

#### **Test Scenario and Test Cases**

The terms 'test scenario' and 'test cases' are used interchangeably; however, there is a difference between the two. To put it simply, a test scenario is a high-level concept of ‘what to test,’ whereas, a test case is a detailed procedure for ‘how to test.’ A single test scenario may consist of many test cases. The following comparison between the two will help you in clarifying the meaning of the two terms:

|  |  |
| --- | --- |
| **Test Scenario** | **Test Cases** |
| A one-line statement that notifies what area in the application will be tested. | Test cases are written to keep track of the testing coverage of a software. |
| Each test scenario has many test cases associated with it. Test scenario means making a linear statement about what to test separately for each test case. | A test case means documenting, in detail, the pre-conditions, how to test, and the expected results etc. |
| Test Scenario means talking and thinking over requirements, in detail, so that a procedure for each test case can be written separately | Test cases involve a series of steps, conditions, and inputs that will be used while performing the test. |
| Test scenarios are used for ensuring that testing of all the process flows is done from end to end. | A detailed test case document is a fool-proof guard for new software tester. |
| Scenario testing means to check how an application works by putting you in the shoes of an end user. The scenario tester needs to take help from the client, stakeholder, or the developers. | When the development is happening onsite and QA is happening Offshore, it is the test case that helps in understanding and making both developer and QA in sync with one another. |

A ‘Test Case’ consists of the following components:

* Test case ID
* Product module
* Product version and revision history
* Purpose
* Assumptions and Pre-conditions
* Steps
* Expected outcome
* Actual outcome
* Post-conditions

#### **Traceability Matrix**

A Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table with the purpose to trace requirements. The RTM document has the following goals:

* To make sure the software is developed as per the mentioned requirements. Each requirement in the RTM needs to be linked to its associated test case.
* To find the root cause of any bug.
* To trace the developed documents during different phases of SDLC.

***Types of Testing***

***5.1 Manual Testing***

Manual testing, as the name specifies, is software testing performed manually and not by any automated tools or a script.

* It is a kind of test where the tester takes over the role of an end-user to identify any unexpected bug or behavior.
* The sequence in which manual testing is performed: unit testing, integration testing, system testing, and user acceptance testing.
* Test plans, test cases, and test scenarios are crucial for ensuring the completeness of testing.

***5.2 Automation Testing***

Automation testing or Test Automation, is the process when a tester writes scripts to see whether the execution of a test-case is perfect or not, or uses another software to test the finished product.

Automation Testing is used for re-running the test scenarios that were performed manually earlier.

Automation testing has the following advantages:

• Quick and can be performed repeatedly in a short period of time

• Ideally used for regression testing

• Tests the application from load, performance, and stress points of view

• Increases the test coverage

• Improves accuracy

• Saves time and money

***5.3 Software Testing Tools***

The following tools are generally used for automation testing:

• HP Quick Test Professional

• Selenium

• IBM Rational Functional Tester

• Silk Test

• Test Complete

• Testing Anywhere

• Win Runner

• Load Runner

• Visual Studio Test Professional

## **Testing Levels**

In the software development process, tests are frequently grouped by where they are added, or by the level of specificity of the tests. Testing, divided into different levels, is required to identify the missing areas between the development life cycle phases and also to prevent overlap and repetition.

Testing can be divided into two basic levels:

1. Functional Testing
2. Non-functional Testing

***Functional Testing***

Functional testing is, essentially, a kind of compliance testing conducted on a complete, integrated system with respect to specificity and requirements.

The below five steps are generally involved in the functional testing:

1. Determination of the functionality
2. Creation of test data
3. Defining the output
4. Writing the test scenarios and the execution of test cases
5. The comparison of actual and expected results based on the executed test cases

Functional Testing has been divided into the 9 following levels:

* 1. **Unit Testing**

The goal of unit testing is to isolate the program into its individual parts and see that each is correct in terms of requirements and functionality. It is done by the developers on a piece of code they have written for a class, function, procedure or interface.

* 1. **Component Testing**

At this level, the testing of each component or module in an application is done separately. If an application contains 5 components, each component is tested separately and efficiently before proceeding to the next step.

* 1. **Component Integration Testing**

The testing of 2 combined modules of an application to determine if they function correctly is known as Component Integration Testing. Component Integration Testing is done to ensure that the code does not break after the components are put together.

* 1. **System Integration Testing:**

Through this testing type, the testers test that all the related systems maintain data integrity and operate in coordination with other systems in the same environment.

* 1. **System Testing**

System testing basically means to test the compatibility of the application with the system.

* 1. **Regression Testing**

Regression testing is a type of system testing conducted to ensure that changes (fixing of defects or enhancements) to the software have not adversely affected its functionality. If there are no new test cases for regression testing, the tests are conducted by re-executing the previous test cases.

Though regression testing can be performed during any level of testing (Unit, Integration, System, or Acceptance), it is ideally performed during the system testing level.

Due to the importance of regression testing, regression test automation tools are adopted by most companies.

* 1. **Acceptance Testing:**

This type of testing is conducted by the QA team to gauge if the application meets all the intended specifications and if it satisfies all the requirements of a client.

* 1. **Alpha Testing:**

This test is performed at the end of the development process and is done amongst the team members. Alpha testing combines the unit testing, integration testing, and system testing.

* 1. **Beta Testing:**

Beta testing means testing the application at the client or customer’s site by a sample of the intended audience. This testing is also known as pre-release testing.

## **Non-Functional Testing**

#### **Performance Testing:**

Performance testing is mostly used in the identification of any bottlenecks or performance issues rather than for finding bugs in the software.

1. **Load Testing:**

This process is utilized in judging the behavior of a software when it is put under maximum load in terms of the number of people accessing the software or a large amount of input data.

1. **Stress Testing:**

Stress testing judges the behavior of a software under abnormal conditions like random shutting down, a restart of the network ports, or access to the database getting off etc.

1. **Usability Testing:**

Usability testing is a black-box technique that serves to identify any error(s) or improvements in the software observed by testers through the viewpoint of the user.

1. **Security Testing:**

Security testing involves testing a software in order to identify any flaws and gaps from a security and vulnerability point of view.

1. **Portability Testing:**

Portability testing includes testing a software with the aim to ensure its reusability and portability to another computer.

## **Testing Methods**

***Black-Box Testing***

Black-box testing or behavioral testing is a testing that checks the functionality of the software without looking at its internal code structure, implementation details, or knowledge of its internal paths.

The objectives of this testing is to check the software with respect to the requirements and specifications. The testing focuses on providing inputs and executing outputs without knowing where the inputs are getting worked upon.

Tests are completed from an end-user's point of view, because he/she is the one who would be accepting or rejecting the system. For this reason, the testing is also referred to as acceptance testing.

Advantages of black box testing:

* When used on large systems, it is quite efficient
* Black box tests are reproducible
* Since the tester is isolated from the developer, testing is balanced and unprejudiced
* Even a non-technical person is eligible for testing
* The testing helps to identify any vagueness and contradiction in functional specifications
* Test cases can be designed as soon as the functional specifications are complete
* It also tests the environment of the program running
* The invested effort can be used multiple times

Disadvantages of black box testing:

* Results are often overestimated
* Not all properties of a software product are tested
* The reason for a failure cannot be known
* In absence of functional specifications, test cases are challenging to design
* Tricky inputs cannot be designed
* There is a possibility of unidentified paths occurring during the testing process, which cannot be identified
* There is a chance of repeating tests already performed by the developers

## **White-Box Testing**

Also known as code-based testing or structural testing, white-box testing validates a software application’s mechanisms, internal framework, objects, and components. This testing variation follows a methodology that not only verifies the design specifications of the software, but also highlights its vulnerabilities. Since the software’s internal mechanisms are clearly reflected through this testing, it is also known as a clear box, glass box, and transparent box testing.

During this testing, the code is made to run with preselected input values and is validated against preselected output values. If a mismatch is detected in the process, a possibility of a bug is highlighted. Each statement, object, and function is tested individually. This testing also involves the writing of software code stubs and drivers.

#### **Advantages of white-box testing**

* Each statement, object, and function is tested individually to achieve effective and through testing of an application
* This methodology is useful for component integration, system testing, and to achieve code optimization
* Enables the identification of errors in code syntax, logic, data flow, and condition
* Helps to describe any new implementation
* Test cases can be easily automated
* Provides a clue when to stop the testing

#### **Disadvantages of black box testing**

* Since this type of testing requires a skilled tester, it is costly
* Looking into every bit of code occasionally increases complications, resulting in the failure of the application
* Updated tests scripts are required every time the implementation is changed
* If the code is large, it can be very time consuming

## **Grey-Box Testing**

A combination of white-box testing and black-box testing is known as grey box testing. Grey-box testing for an application is done with limited knowledge of its internal working. The tester has access to design documents and the database, but not to the source code. Grey-box testing is performed by end-users as well as by testers and developers.

#### **Advantages of Grey-box testing**

* It offers the combined benefit of both white-box and black-box testing
* Grey-box testers rely on interface definition and functional specifications instead of source code
* Excellent test scenarios around communication protocols and data type handling can be designed
* The testing will be performed from the user point of view instead of designer
* Testing is done on the basis of high-level database diagrams and data flow diagrams

***Disadvantages of Grey-box testing***

* Provides limited test coverage in the absence of access to source code
* If the software designer has already run a test case, there is a chance of tests becoming redundant
* Testing is not exhaustive
* It is not suited for algorithm testing

## **Version Control Systems**

As a category of software tools, ‘Version Control Systems’ or the Source Code Management System helps a software team in managing the changes to the source code over a period of time. This software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code. This revision control system helps to fix mistakes and, at the same time, also minimizes the disruption to all team members.

The version control system utilizes a mechanism for keeping multiple versions of a file, so when a file is modified, it is available in a track change mode to all the team members working in parallel. The system also provides access to all the previous revisions.

Dating back to the 1970s, the proprietary UNIX tool called SCCS (Source Code Control System) had been the first popular version control system. This was superseded by RCS(Revision Control System), and later by CVS(Concurrent Versions System). Subversion and Git are the most popular version control systems available today.

#### **Purpose of the Version Control System**

Version control is used in preference over the source control because version control is not just for source code. Ideally, every single artifact related to the creation of the software should be under version control as given below:

* Software development environment
* Database scripts
* Build and deployment scripts
* Documentation
* Libraries and configuration files
* DNS Zone Files
* Firewall Configuration etc.
* Compiler and collection of tools etc.

All of these artifacts compliment the development and deployment process. It is advisable to store all of these items in the versioning control system. Having all this information at hand makes it easier to re-create the testing and production environments for an application to run.

***Features of a Good VCS***

* **Backup and Restore:** Files are saved as they are edited, and you can jump to any moment in time.
* **Synchronization**: Helps the team members share files so that every member stays up-to-date with the latest version.
* **Short-term undo:** In case some error occurs, the short-term undo helps to remove the changes and go back to the “last known” acceptable version in the database.
* **Long-term undo:** When a change a year or more ago created a bug. The long-term undo allows you to jump back to the old version to see what changes were made on that day.
* **Track Changes**: Since all files are updated, a message or comment can be made to explain why the change happened (stored in the VCS, not the file). This provides a means to see how a file has evolved over time, and why.
* **Track Ownership**: As the VCS tags every change with the name of the person who made it, it can be very helpful in ownership tracking .
* **Sandboxing**: If you want to make a large change, you can do it in an isolated area and test/work-out the kinks before “checking in” your changes.
* **Branching and merging**: You can also branch a copy of your code into a separate area and modify it in isolation (tracking changes separately) that can be merged back into the common area.

#### **Important Concepts/Terms**

The following concepts are used by different version control systems:

**Basic Setup**

The database storing the files: Repository (repo)

The computer storing the repo: Server

The computer connecting to the repo: Client

Your local directory of files: Working Set/Working Copy

where you make changes:

If the code is a family tree: Trunk/Main

(The primary location for code in the repo)

#### **Basic Commands or Actions**

* Add: Put a file into the repo for the first time, i.e. begin tracking it with Version Control.
* Revision: What version a file is on (v1, v2, v3, etc.).
* Head: The latest revision in the repo.
* Check out: Download a file from the repo.
* Check in: Upload a file to the repository (if it has changed).
* Check in Message: A short message describing what was changed.
* Change log/History: A list of changes made to a file since it was created.
* Update/Sync: Synchronize your files with the latest from the repository. This lets you grab the latest revisions of all files.
* Revert: Throw away your local changes and reload the latest version from the repository.

#### **Advanced Actions**

* **Branch**: Create a separate copy of a file/folder for private use (bug fixing, testing, etc). Branch is both a verb (“branch the code”) and a noun (“Which branch is it in?”).
* **Diff/Change/Delta**: Finding the differences between two files. Useful for seeing what changed between revisions.
* **Merge (or patch)**: Applying the changes from one file to another to bring it up-to-date. For example, you can merge features from one branch into another (at Microsoft this was called Reverse Integrate and Forward Integrate).
* **Conflict:** When pending changes to a file contradict each other (both changes cannot be applied).
* **Resolve**: Fixing the changes that contradict each other and checking in the correct version.
* **Locking**: Taking control of a file so that nobody else can edit it until you unlock it. Some version control systems use this to avoid conflicts.
* **Breaking the lock**: Forcibly unlocking a file so that you can edit it. This may be needed if someone locks a file and goes on vacation (or “calls in sick” the day Halo 3 comes out).
* **Check out for edit**: Checking out an “editable” version of a file. Some VCS’ have editable files by default, others require an explicit command.

***References***:

1. <https://www.tutorialspoint.com/software_testing/software_testing_tutorial.pdf>
2. <https://www.guru99.com/software-testing.html>
3. <http://www.mosaicinc.com/mosaicinc/rmThisMonth.asp>
4. <https://www.techopedia.com/definition/16373/debugging>
5. <http://www.softwaretestingclass.com/what-is-difference-between-test-cases-vs-test-scenarios/>
6. <http://www.mathematik.uni-ulm.de/~melzer/thesis/node27.html>
7. <https://www.codeproject.com/Articles/5579/Black-Box-Testing-Its-Advantages-and-Disadvantages>
8. <https://www.invensis.net/blog/it/white-box-software-testing-advantages-disadvantages/>
9. <https://www.zyxware.com/articles/4983/advantages-and-disadvantages-of-grey-box-testing>
10. <https://reqtest.com/testing-blog/differences-between-the-different-levels-of-tests/>
11. <http://istqbexamcertification.com/what-are-software-testing-levels/>
12. <https://betterexplained.com/articles/a-visual-guide-to-version-control/>

# Module 11

## **Cloud Computing**

#### **Overview of Cloud:**

Often referred to simply as “the cloud,” Cloud Computing basically means the on-demand delivery of a range of computing resources — everything from applications to data centers — over the internet on a pay-for-use basis.

The cloud is something that is present at a remote location which provides services on public or private networks (i.e., WAN, LAN or VPN).

Applications such as email, web conferencing, customer relationship management (CRM), etc. all run in cloud.

Cloud computing has the following main characteristics:

* **Elastic Resources**: A cloud contains resources that can be scaled up or down quickly to meet the demand
* **Metered Service**: The service charges you pay only for what you use
* **Self Service**: The cloud provides self-service access to all the resources it contains

#### **Details of Cloud Computing**

Cloud computing comprises the whole gamut of manipulating, configuring, and accessing the applications online, thereby avoiding the need for installing software on your local PC.

Special Features of Cloud Computing:

* Applications are assessed as utilities, over the Internet.
* Applications can be manipulated and configured any time of the day.
* One does not require installing a specific software to access or manipulate a cloud application.
* Cloud computing also offers online development and deployment tools, and also the programming runtime environment. Cloud resources provide platform independent access to different types of clients. The resources can be used without any interaction with the cloud service provider.

## ***Benefits of Cloud Computing***

1. ***Cost***

Since cloud computing eliminates the capital expense of buying hardware, software, setting up data centers, servers, and round-clock electricity for power and cooling, it is quite cost-effective.

1. ***Speed***

Since cloud computing provides self-service on demand, large amounts of computing resources can be provisioned in minutes, just with a few mouse clicks. Thus, business developers save time so that they can divert their efforts towards capacity planning.

1. ***Global scale***

Another benefit of cloud computing is its flexibility. Cloud computing helps you receive just the right amount of IT resources— computing power, storage, bandwidth etc. whenever it is needed, and also from the right geographic location.

1. ***Productivity***

Cloud computing removes the need for many time-consuming management jobs such as software patching so that the IT teams can better utilize their time for achieving important business objectives.

1. ***Efficiency in Performance***

The hardware required for running a worldwide network of secure data centers is regularly updated so cloud computing services provide greater efficiency and better overall performance.

1. ***Reliability***

Since cloud computing provides data backup at multiple redundant sites, disaster recovery and business continuity has become easier and less expensive.

**Cloud Computing Service Models**

The cloud computing services can be divided into three broad categories. As these services build on top of one another, they are often called the ‘cloud computing stack’. It becomes easier to accomplish business goals once it is understood what they are and how their services are different from each other.

* **Infrastructure as a Service (IaaS):** IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc. on a pay-as-you-go basis.
* **Platform as a Service (PaaS):** PaaS provides the runtime environment for applications, development & deployment tools, etc. making it easier for developers to quickly create web or mobile apps without the hassle of setting up or managing the infrastructure of servers, storage, networks, and databases.
* **Software as a Service (SaaS**): This service delivers software applications on-demand over the Internet, typically on a subscription basis. Cloud providers host and manage software applications and the underlying infrastructure along with their maintenance/upgrades and security patching.

**Cloud Computing Deployment Models**

There are four types of access provided by cloud computing:

#### **The Public Cloud**

The Public Cloud allows accessibility of systems and services to the general public. Public clouds can be less secure because of its openness (e.g. e-mail).

Public clouds are owned and operated by companies offering rapid access over a public network. With public cloud services, users do not need to purchase hardware, software, or supporting infrastructure; instead, everything is owned and managed by the providers.

#### **Key aspects of public cloud**

1. Innovative SaaS business apps can be provisioned for applications ranging from customer resource management (CRM) to transaction management and data analytics.
2. Flexible, scalable IaaS for storage and computer services on a moment’s notice.
3. Powerful PaaS for cloud-based application development and deployment environments.

#### **Private Cloud**

The Private Cloud allows systems and services to be accessible only within an organization and, therefore, offers increased security.

A private cloud can be managed internally or by a third party, and can also be hosted either internally or externally. Private clouds provide cloud efficiencies, while providing more control of resources which saves you the hassle of multi-tenancy.

#### **Key aspects of private cloud**

1. A self-service interface of a private cloud allows the IT staff to quickly provision, allocate, and deliver on-demand IT resources.
2. Provides highly automated management of resource pools for storage, analytics, and middleware.
3. Sophisticated security and governance designed for a company’s specific requirements.

The Community Cloud allows systems and services to be accessible to a group of organizations.

#### **Hybrid Cloud**

The Hybrid Cloud is a mixture of public and private cloud, where the critical activities are performed using private cloud and the non-critical activities are performed by a public cloud. A hybrid cloud uses a private cloud foundation strategically integrated with public cloud services. Since a private cloud cannot exist in isolation with the rest of a company’s IT resources and the public cloud, most companies opt for hybrid clouds.

#### **Key aspects of hybrid cloud**

1. Allows companies to keep critical applications and sensitive data in a traditional data center environment on a private cloud.
2. Offers the advantage of public cloud resources like SaaS, for the latest applications, and IaaS, for elastic virtual resources.
3. Facilitates portability of data, apps and services and more choices for deployment models.

#### **How Cloud Computing Works?**

Most cloud computing services work a little differently depending upon who the provider is; however, all cloud services provide a friendly, browser-based dashboard through which IT professionals and developers can easily order resources and manage their accounts. Some cloud computing services are also designed to work with REST APIs and a command-line interface (CLI), giving developers multiple options.

#### **Cloud Computing-Technologies**

There are certain technologies working behind the cloud computing platforms that make cloud computing flexible, reliable, and usable. These technologies are listed below:

* **Virtualization:** This technique allows the sharing of a single physical instance of an application or resource among multiple organizations or tenants (customers).
* **Service-Oriented Architecture (SOA)**: SOA is a technique that helps in using applications as a service for other applications. This service is independent of the type of vendor, product or technology.
* **Grid Computing**: Grid Computing means a type of distributed computing through which computers from multiple locations are connected with each other. The developers, therefore, can work as a group towards achieving the common objectives.
* **Utility Computing**: Utility computing is based on a pay-per-use model. It offers computational resources on demand as a metered service.

## **Cloud Computing-Architecture**

The Front End refers to the client part of a cloud computing system. This consists of interfaces and applications that are required to access the cloud computing platforms, e.g., Web Browser. On the other hand, Back End refers to the cloud and all the resources required to provide various services.

#### **Salient Points**

* The back end ensures built-in security mechanism, traffic control and protocols.
* Certain protocols, known as middleware, are employed by the server which helps the connected devices to communicate with each other.

#### **Understanding Cloud Computing Service Models**

Cloud computing offers a model to enable a ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). These resources can be rapidly provisioned and released without any interaction of the service provided and minimal management efforts.

## **Infrastructure as-a-Service (IaaS)**

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc. on a pay-per-use basis.

Apart from these resources, the IaaS also offers:

* Virtual machine disk storage
* Virtual local area networks (VLANs)

#### **Load balancers**

* IP addresses
* Software bundles

As the name suggests, all of the above resources are made available to the end user via server virtualization.

#### **Special Characteristics**

Here are some important characteristics of IaaS service model:

* Offers virtual machines with pre-installed software
* Offers virtual machines with pre-installed Operating Systems such as Windows, Linux, and Solaris
* Provides an on-demand availability of resources
* Allows storing of copies of particular data in different locations
* The computing resources can be easily scaled up and down

#### **Benefits of IaaS**

Through IaaS, the cloud provider is able to freely locate the infrastructure over the Internet offering the following benefits:

* It is cost effective since it saves you from investing in your own hardware
* The infrastructure can be scaled up on-demand to support dynamic workloads
* Flexible, innovative services like renting of computing hardware available on demand
* Full Control of the computing resources through administrative access to VMs
* Portability and interoperability with Legacy Applications

#### **Platform-as-a-Service (PaaS)**

PaaS offers the runtime environment for applications to support the complete lifecycle of building and delivering web-based (cloud) applications along with the required development & deployment tools. These tools (point-and-click etc.) enable non-developers to create web applications.

Google's App Engine and Force.com are some examples of PaaS offering vendors.

#### **Benefits of PaaS model:**

* **Reduces time:** The time required for developing the web-based applications is greatly reduced.
* **Reduces administrative overhead**: You do not have to bother the administration since it is the responsibility of the cloud provider.
* **Reduces cost of purchase**: There is no need to purchase expensive hardware, servers, power, or data storage.
* **Offers scalable resources**: It gets scaled up or down automatically based on application resource demands.
* **Availability of latest system software**: The cloud provider maintains software versions and patch installations.
* **Reduces complexity**: Middleware as a service reduces the complexity of database elements and operating systems.

#### **Characteristics**

PaaS offers a browser-based development environment. The developer is able to create a database and edit the application code either via Application Programming Interface or point-and-click tools.

* Provides built-in security, scalability, and web service interfaces.
* Provides built-in tools for defining workflow, approval processes, and business rules.
* The integration with other applications on the same platform is easily possible. Its’ web services interface and also allows connecting to the applications outside the platform.

**Software-as-a-Service (SaaS)**

Through SaaS model, any end user can sign up and rapidly start using innovative business applications that are deployed on a hosted service and are accessible via the Internet.

The following are some examples of SaaS applications:

* Human Resource (HR) Solutions
* Billing and Invoicing System
* Customer Relationship Management (CRM) applications
* Help Desk Applications

Though some of the SaaS applications such as an Office Suite are not customizable through an Application Programming Interface (API), the developer is able to develop customized applications for organizations.

#### **Benefits**

Besides the benefits in terms of scalability, efficiency, and performance, SAAS also offers the following advantages:

* Modest Software Tools
* Efficient use of Software Licenses
* Centralized Management & Data
* Platform responsibilities managed by provider
* Multi-Tenant solutions
* Easy accessibility of apps from any connected computer
* Data backup is available

#### **Important Characteristics**

* All SaaS software is available over the Internet
* The software is maintained by the vendor
* The license to the software may be subscription-based or usage-based which is billed on a recurring basis
* SaaS applications are cost-effective since they do not require any maintenance
* Quick on-demand availability and scaling facility
* Automatically upgraded and updated
* SaaS also offers a ‘share data model’ so multiple users can share a single instance of infrastructure. All users are running the same version of the software

#### **References**

1. <https://www.tutorialspoint.com/cloud_computing/cloud_computing_tutorial.pdf>
2. <https://www.ibm.com/cloud/learn/what-is-cloud-computing>
3. <https://www.bluepiit.com/blog/different-types-of-cloud-computing-service-models/>
4. <https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/>