OFFICE AUTOMATION

UNIT I

Problem definition and Analysis:

Problem definition

A clear problem definition is the first, and, perhaps, mostimportant step toward rationally selecting the best alternative. Many dedicated and intelligent individuals have produced elegant solutions for problems other than those they were tasked to solve. Therefore, a good executive decision maker participates in problem definition because this step establishes the goal for everything else that follows and places a premium on professional judgment. we will discuss the opening phase of our Executive Decision-Making Framework: we will examine techniques to describe defence problems in terms that are meaningful to our organization and our decision maker, examine those problems' contexts and boundaries, and then prepare for the Analysis Phase by specifying its objectives.

The First Four Steps

1. Collect and analyze information and data
2. Talk with people familiar with the problem
3. If at all possible, view the problem first hand
4. Confirm all findings

Problem Analysis

- Definition: the process of understanding the real-world problems and users needs and proposing abstract solutions to those problems.
- Goal: gain a better understanding, before development begins, of the problem to be solved.
- Avoid to jump to conclusions by identifying the root cause of the problem.
- Identify the sources of information for system analysis.

Step1: Gain agreement on the problem

Definition

- Write a simple and clear definition of the problem

Description

- Establish an order of importance for all features of the system.

Step 2 : Identify the root causes of the problem

- Make sure that the problem identified is the real Problem.
- Sometimes, a problem hides other more important Problems.
- Addressing the wrong problem leads to failure.

<u>Problem solving techniques:</u>

Problem solving technique is a set of techniques and graphical tools that helps in providing logic for solving a problem. These tools are used to express the logic of the problem by specifying the correct sequence of all instructions to be carried out

There are three important problem solving tools used for program development. They are:
i. Algorithms
ii. Flowcharts
iii. Pseudo codes

<u>Algorithms:</u>

An algorithm is a set of instructions, sometimes called a procedure or a function that is used toperform a certain task. This can be a simple process, such as adding two numbers together, ora complex function, such as adding effects to an image. For example, in order to sharpen a digital photo, the algorithm would need to process each pixel in the image and determine whichones to change and how much to change them in order to make the image look sharper

Algorithms are one the most basic tools that are used to develop the problem solving logic. An algorithm is defined as a finite sequence of explicit instruction that when provided with a set of input values process an output and then terminates. Algorithms can have steps that repeat or require decisions until the task is completed.

Example: To find out the largest number among three numbers A, B & C.

Step 1: Start

Step 2: Read three Numbers A, B & C.

Step 3: Find the largest number between A & B and store it in Max_AB.

Step 4: Find the larger number between MAX_AB and C and store it in MAX.

Step 5: Display MAX.

Step 6: Stop.

Properties:

1. There must be no ambiguity in any instruction.
2. There should not be any uncertainty about which instruction is to be executed next.
3. The algorithm should conclude after a finite number of steps.
4. The algorithm must be general enough to deal with any contingency.

Advantages and Disadvantages of Algorithms

Advantages

The use of algorithms provides a number of advantages. One of these advantages is in the development of the procedure itself, which involves identification of the processes, major decision points, and variables necessary to solve the problem. Developing an algorithm allows and even forces examination of the solution process in a rational manner. Identification of the processes and decision points reduces the task into a series of smaller steps of more manageable size. Problems that would be difficult or impossible to solve wholesale can be approached as a series of small, solvable subproblems. The required specification aids in the identification and reduction of subconscious biases. By using an algorithm, **decision-making** becomes a more rational process

Disadvantages

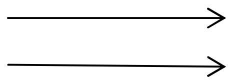
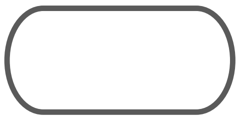
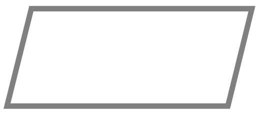
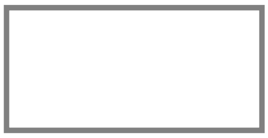
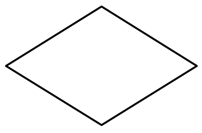
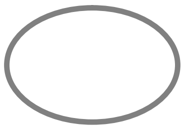
One disadvantage of algorithms is that they always terminate, which means there are some computational proceduresoccasionally even useful oneswhich are not algorithms. Furthermore, all computational procedures, whether they terminate or not, can only give computable results,so you cannot, for example, design a program which determines a busy beaver number more quickly than could be done by actually running the associated types of turing machines.

Flowchart:

A flowchart is a pictorial representation of an algorithm in which the steps are drawn in the form of different shapes of boxes and the logical flow is indicated by interconnecting arrows. The boxes represent operations and the arrows represent the sequence in which the operations are implemented.

Flowchart Symbols:

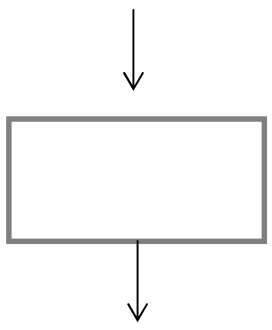
Symbol Name Description

Symbol	**Symbol Name**	**Description**
	Flow Lines	Used to connect Symbols
	Terminal	Used to represent start, pause or halt in the program logic.
	Input/Output	Represents the information entering or leaving the system.
	Processing	Represents arithmetic and logical instructions.
	Decision	Represents a decision to be made
	Connector	Used to join different flow lines.

Guidelines for preparing flowcharts:

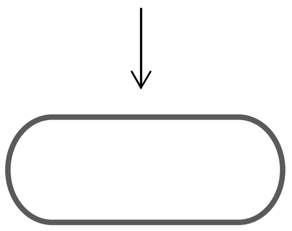
1. The flowchart should be clear, neat and easy to follow.
2. The flowchart must have a logical start and finish.

3. Only one flow line should come out from a process symbol.



4. Only one flow line should enter a decision symbol. However, two or three flow lines may leave the decision symbol.

5. Only one flow line is used with a terminal symbol.



6. Within standard symbols, write briefly and precisely.

7. Intersection of flow lines should be avoided.

8. It is useful to test the validity of the flowchart with normal/unusual test data.

Benefits of Flow charts: A flowchart helps to clarify how things are currently working and how they could be improved. It also assists in finding the key elements of a process by drawing clear lines between the end of one process and the start of next one.

Reasons for using flowcharts as a problem solving tool are given below:

i. Makes Logic Clear.

ii. Communication.

iii. Effective Analysis.

iv. Useful in coding.

v. Proper testing and debugging.

vi. Appropriate documentation.

Limitations of Flowcharts: The limitations of flowcharts are:

i. Complex.

ii. Costly.

iii. Difficult to modify.

iv. No update.

PROGRAMMING LANGUAGES

Computer languages are classified into three categories:
1. Machine language
2. Assembly language:
3. High level languages:

MACHINE LANGUAGE

The form in which information is denoted or represented by a combination of “1” and “0”s is called machine language. Programs and data are ultimately represented in machine language in memory and other hardware of the computer.

To start with computer programmers used machine language for programming. It was tedious since each operation that the computer has to perform is denoted by a particular combination of”1” s and”0”s.Also each computer had its own machine language code. One has to remember all these codes to write a program.

While machine language had the drawbacks mentioned above, it was very efficient in using computer circuits. There was no necessity for software to convert since we can directly represent the machine language in the computer circuits without the necessity of conversion.

Advantages of Machine Language:

Even though machine languages are not a human friendly language, it offers following advantages.

a) Translation Free:

Machine language is the only language that computers can directly execute without the need for conversion. Even an application using high- level languages, has to be converted into machine-readable form so that the computer can understand the instructions.

b) High Speed:

Since no conversion is needed, the applications developed using machine language are extremely fast. It is usually used for complex applications such as space control system, nuclear reactors, and chemical processing.

Ø **Disadvantages of Machine Language:**

There are many disadvantages in using machine language to develop programs. Some of these are

a) Machine Dependent:

Every computer type differs from the other, based on its architecture. Hence, an application developed for a particular type of computer may not run on the other type of computer. This any prove costly as well as difficult for the organizations.

b) Complex Language:

Machine language is very difficult to read and write. Since all the data and instructions must be converted to binary code, it is almost impossible to remember the instructions. A programmer must specify each operation, and the specific locations for each piece of data and instructions to be stored. It means that a programmer practically needs to be hardware expert to have proper control over the machine language.

<u>ASSEMBLY LANGUAGE:</u>

As computer field advanced, computer scientists felt the necessity of making computer programming easier. They invented computer languages that are easy to remember and use.

These languages called assembly language uses mnemonic codes. for Example: For adding two quantities X and Y, You may code as follows.

ADD X Y

ADD is called operation code that tells the computer what operation it has to perform. X and Y are called operation addresses. These refer to the memory addresses where the quantities you want to add are stored. X refers to the location of first quantity and Y the second quantity.

These are called symbolic address, since they do not refer to any fixed address. Software called assembler translates this assembly language program into machine language program which you can load into computer memory and execute.

When the program is loaded into memory for execution the symbolic addresses are automatically converted into actual physical addresses of operands involved in an instruction.

Ø Assembly language offers the following advantages:

a) The assembly language uses mnemonic codes, for programming. So it is easier for the programmer to use assembly language than machine language for programming.

b) The assemblers, in addition to translating the assembler language program into machine language, list the program statements (called source code) and also the syntax errors that the programmer committed.

c) An understanding of assembly language provides knowledge of:

- o Interface of programs with OS, processor and BIOS;
- o Representation of data in memory and other external devices;
- o How processor accesses and executes instruction;

d) Other advantages of using assembly language are:

- o It requires less memory and execution time;
- o It allows hardware-specific complex jobs in an easier way;

Ø Assembly language has certain disadvantages too:

a) One has to spend considerable time in writing the assembly language program.

b) An assembly language program has to be converted into machine language program using assembly.

c) While it is easier to use than the machine language one still has to remember the operation codes for various operations.

HIGH LEVEL LANGUAGES:

During 1960s computers started to gain popularity and it became necessary to develop languages that were more like natural languages such as English so that a common user could use the computer efficiently. Since assembly language required deep knowledge of computer architecture, it demanded programming as well as hardware skills to use computers. Due to computer's widespread usage, early 1960s saw the emergence of the third generations programming languages (3GL). Languages such as COBOL, FORTRAN, BASIC, and C are examples of 3GLs and are considered high- level languages.

High level languages are similar to English language. Programs written using these languages can be machine independent. A single high- level statement can substitute several instructions in machine or assembly language. Unlike assembly and machine programs, highlevel programs may be used with different types of computers with little or no modification, thus reducing the re-programming time.

Ø Translating High-level Language to machine language:

Since computers understand only machine language, it is necessary to convert the highlevel language programs into machine language codes. This is achieved by using language translators or language processors, generally known as compliers, interpreters or other routines that accepts statements in one language and produces equivalent statements in another language.

a) Compiler:

A compiler is a kind of translator that translates a program into another program, known as target language. Usually, the term compiler is used for language translator of high- level language into machine language. The complier replaces single high- level statement with a series of machine language instruction. A compiler usually resides on a disk or the storage media.

When a program is to be complied, its complier is loaded into main memory. The compiler stores the entire high- level program, scans it and translates the whole program into an equivalent machine language program. During the translation process, the compiler reads the source program and checks the syntax (grammatical) errors. If there is any error, the compiler generates an error message, which is usually displayed on the screen. In case of errors, the compiler will not create the object code until all the errors are rectified.

b) Interpreter:

Unlike compilers, an interpreter translates a statement in a program and executes the statement immediately, before translating the next source language statement. When an error is encountered in the program, the execution of the program is halted and an error message is displayed. Similar to compilers, every interpreted language such as BASIC and LISP has its own interpreters.

Ø Advantages of high-level languages:

High- level languages (HLL) are useful in developing complex software, as they support complex data structures. It increases the programmer's productivity (the number of lines of code generated per hour). Unlike assembly language, the programmer does not need to learn the instructions set of each computer being worked with. The various advantages of using high- level languages are discussed below:

a) Readability:

Since high- level languages are closer to natural languages, they are easier to learn and understand. In addition, a programmer does not need to be aware of computer architecture; even a common man can use it without much difficulty. This is the main reason of HLL's popularity.

b) Machine Independent:

High- level language are machine independent in the sense that a program created using HLL can be used on different platforms with very little or no change at all.

c) Easy Debugging:

High- level languages include the support for ideas of abstraction so that programmers can concentrate on finding the solution to the problem rapidly, rather than on low- level details of data representation, which results in fewer errors. Moreover, the compilers and interpreters are designed in such a way that they detect and point out the errors instantaneously.

d) Easier to Maintain:

As compared to low-level languages, the programs written in HLL are easy to modify and maintain because HLL programs are easier to understand.

e) Low Development Cost:

High- level languages permit faster development of programs. Although a high- level program may not be as efficient as an equivalent low- level program, but the savings in programmer's time generally outweighs the inefficiencies of the application. This is because the cost of writing a program is nearly constant for each line of code, regardless of the language. Thus, a high- level language, where each line of code translates to 10 machine instructions, costs only a fraction as compared to program developed in a low- level language.

f) Easy Documentation: Since the statements written in HLL are similar to natural languages they are easier to understand as compared to low- level languages.

<u>Ø Disadvantages of high-level languages:</u>

The main disadvantages of this language are:

a) Poor Control on Hardware:

High- level languages are developed to ease the pressure on programmers so that they do not have to know the intricacies of hardware. As a result, sometimes the applications written in high- level languages cannot completely harness the total power available at hardware level.

b) Less Efficient:

The HLL applications are less efficient as far as computation time is concerned. This is because, unlike low- level languages, high- level languages must be created and sent through another processing program known as a complier. This process of translation increases the execution time of an application. Programs written in high- level languages take more time to execute, an require more memory space. Hence, critical applications are generally written in low- level languages.

PROBLEM TESTING DOCUMENTATION:

Testing documentation involves the documentation of artifacts that should be developed before or during the testing of Software.

Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc. This section describes some of the commonly used documented related to software testing such as:

- Test Plan
- Test Scenario
- Test Case

- **Test Plan**

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, and the limitations of the testing and the schedule of testing activities. Typically the Quality
Assurance Team Lead will be responsible for writing a Test Plan.

- **Test Scenario**

It is a one line statement that notifies what area in the application will be tested. TestScenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred
scenarios depending on the magnitude and complexity of the application. The terms 'test scenario' and 'test cases' are used interchangeably, however a test scenario has several steps, whereas a test case has a single step. Viewed from this perspective, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on theoutput from the previous test.

- **Test Case**

Test cases involve a set of steps, conditions, and inputs that can be used while performing testing tasks. The main intent of this activity is to ensure whether a softwarepasses or fails in terms of its functionality and other aspects. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc. Furthermore, test cases are written to keep track of the testing coverage of a software.
Generally, there are no formal templates that can be used during test case writing.Many test cases can be derived from a single test scenario. In addition, sometimes multiple test cases are written for a single software which are collectively known as test suites.

Testing: It involves identifying bug/error/defect in a software without correcting it.Normally professionals with a quality assurance background are involved in bugsidentification. Testing is performed in the testing phase.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.There are different methods that can be used for software testing.

➢ **Black-Box Testing**

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

➢ **White-Box Testing**

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

DATA PROCESSING

Data processing is the act of handling or manipulating data in some fashion. Regardless of the activities involved in it, processing tries to assign meaning to data. Thus, the ultimate goal of processing is to transform data into information.

Data Processing Concepts

Ø **Data**

Data means any collection of raw hand figures facts. Data can be considered as the raw material of information. The data may be numerical such as payroll, employee Number, etc. or non- numerical like Student names, Product names, etc.

Ø **Data Processing**

As data is in its raw form it cannot solve any problem. The data needs some processing to make it useful. Data processing is the conversion of data into a more useful form. That is, transmission of data into meaningful information is called data processing.

Ø **Information**

The result obtained by data processing is called information. That is, the processed data is known as information. However, information produced in one data processing step may be used as data in the next data processing step.

Data processing may involve various processes, including:

- Validation – Ensuring that supplied data is correct and relevant.
- Sorting – "arranging items in some sequence and/or in different sets."
- Summarization – reducing detail data to its main points.
- Aggregation – combining multiple pieces of data.
- Analysis – the "collection, organization, analysis, interpretation and presentation of data."
- Reporting – list detail or summary data or computed information.
- Classification – separates data into various categories.

Steps in data processing

1. Identifying the data
2. Input of Data.
3. Manipulation of Data.
4. Output of Information.
5. Storage of Information

1. Identifying the data

Accuracy of information depends on accurate data input. The first step in data processing therefore, is to locate necessary facts and figures from source documents. Accurate, relevant and adequate data must be used as input.

2. Input of data:

After extracting the necessary data from the source documents, they must be transposed in a suitable form acceptable to the computer. Great care should be taken to avoid wrong entries in the forms.

3. Manipulation of data:

It involves the process of shifting, sorting and rearranging the given input. Before processing, validation procedures may be built in to the code to so that input forms do not accept any incorrect data.

4. Output of information:

The main purpose of data processing is to provide meaningful information to the decision-maker. Hence, in data processing the person involved must be very careful about what information is needed and in what form he likes to have it.

5. Storage of information:

The data processed need to be kept for future use. All the processed data will need some form of secondary storage. When storing the data, it is always important to maintain a backup. It should be noted that at each and every step the storage might be done.

FILES AND RECORDS

Record

A record is a collection of related data items or fields. Each record normally corresponds to a specific unit of information. For example, various fields in the record.

File

The collection of records is called a file. A file contains all the related records for an application. Files are stored on some medium, such as floppy disk, magnetic tape or magnetic disk.

Database

The collection of related files is called a database. A database contains all the related files for a particular application.

Variable and Fixed Length Records

Records can be of fixed or variable length as,

- **Fixed Length Records**

In this case, all the records in a file have the same number of bytes. Such a file is called a flat file. If all the records are expected to contain essentially the same quantity of data, then fixed length records are used.

- **Variable Length Records**

In this case, records vary in length. Use of variable length records conserves storage space when the quantity of information, of various records in a file, differs significantly.

Logical Versus Physical Record

A logical record contains all the data related to a single entity. It may be a payroll record for an employee or a record of marks secured by a student in a particular examination. A physical record refers to a record whose data fields are stored physically next to one another. It is also the amount of data that is treated as asingle unit by the input-output device. Portions of the same logical record may be located in different physical records or several logical records may be locatedin one physical record.

Definitions:

Record = a collection of related fields.
Field = the smallest logically meaningful unit of information in a file.
Key = a subset of the fields in a record used to identify (uniquely, usually) the record.

A **record** is the collection of fields that relate to a single entity. For example, we could havea student record that includes fields for the student's name, address, homeroom, date of birth, etc. A product record could include fields for the serial number, description, cost price, quantity in stock, etc.

A **file** is a collection of related records. For example, a student file might include all of the records of students enrolled at a school. A police department might keep a file of criminal records, which includes details about all known criminals. Files are stored on secondary storage devices such as hard disks, CD-ROMs etc.
Within a file, all records usually have the same structure. That is, every record in the file contains the same fields. Only the data stored in the fields of different record will be different.