2.1. Problem Analysis

The Seven Steps of Problem Analysis

- **Read the case thoroughly:** To understand fully what is happening in a case, it is necessary to read the case carefully and thoroughly. You may want to read the case rather quickly the first time to get an overview of the industry, the company, the people, and the situation. Read the case again more slowly, making notes as you go.
- **Define the central issue:** Many cases will involve several issues or problems. Identify the most important problems and separate them from the more trivial issues. After identifying what appears to be a major underlying issue, examine related problems in the functional areas (for example, marketing, finance, personnel, and so on). Functional area problems may help you identify deep-rooted problems that are the responsibility of top management.
- **Define the firm's goals:** Inconsistencies between a firm's goals and its performance may further highlight the problems discovered in step 2. At the very least, identifying the firm's goals will provide a guide for the remaining analysis.
- **Identify the constraints to the problem:** The constraints may limit the solutions available to the firm. Typical constraints include limited finances, lack of additional production capacity, personnel limitations, strong competitors, relationships with suppliers and customers, and so on. Constraints have to be considered when suggesting a solution.
- **Identify all the relevant alternatives:** The list should all the relevant alternatives that could solve the problem(s) that were identified in step 2. Use your creativity in coming up with alternative solutions. Even when solutions are suggested in the case, you may be able to suggest better solutions.
- **Select the best alternative:** Evaluate each alternative in light of the available information. If you have carefully taken the proceeding five steps, a good solution to the case should be apparent. Resist the temptation to jump to this step early in the case analysis. You will probably miss important facts, misunderstand the problem, or skip what may be the best alternative solution. You will also need to explain the logic you used to choose one alternative and reject the others.
- **Develop an implementation plan:** The final step in the analysis is to develop a plan for effective implementation of your decision. Lack of an implementation plan even for a very good decision can lead to disaster for a firm and for you. Don't overlook this step. Your teacher will surely ask you or someone in the class to explain how to implement the decision.

2.2. Requirements Analysis

The members of a software development team must have a clear understanding of what the software product must do.

The first step is to perform a thorough analysis of the client's current situation, careful to define the situation as precisely as possible.

This analysis may require examination of a current manual system being operated, or may need an appraisal of some computerized system to be performed. Once a clear picture of the current situation is obtained, then the question of "What must the new product be able to do?" may be answered.

During the requirements phase the developer must :-

- Attempt to identify problems in the current system
- Avoid blindly taking statements about the client's wants e.g. a wish list
- Attempt to determine the real needs of the client
- Recognize the client is not always conscious of all the needs
- Overcome any lack of computer-literacy on the part of the client i.e. bridge the technical divide between developer and client
- Correctly interpret client's requests even if not stated in the best way possible

Requirements analysis begins with the requirements team meeting with members of the client organization. The initial meeting may be used to plan subsequent interviews or techniques for soliciting the relevant information from the client's organization.

Several techniques can be employed to elicit requirements:

- Interviews
- Questionnaires
- Client Documents
- Scenarios
- Rapid Prototypes

Interviews

Both the structured and unstructured interview may be planned. In the case of the structured interview a list of specific closed ended questions are posed and the answers recorded. E.g. “How long does it take to perform activity X” or “How many people are in the marketing dept?” or “How much was spent on the current system last month?”

These type of targeted questions seek information the interviewer deems relevant in finding the true needs of the client.

In the case of the unstructured interview open-ended questions are asked which allow the interviewee to outline broad areas or express views/opinions/convictions that may be hard to quantify. E.g. “Explain why the current product is unsatisfactory?” or “What are the best features of the current system?” or “What would be the most effective way to accomplish task X?”

At the end of the interview process the interviewer prepares a written report outlining the results of the interview. Those interviewed should be given copies and allowed to add/clarify statements made.

Questionnaires

A well-structured questionnaire is a useful tool for gathering information. In the case of large departments/organizations it may be impractical to conduct numerous interviews. Unlike the interview process that is interactive in nature, there is no way to pose new questions (follow-ups) based on the answers given to previous questions. This means a skillful and methodical interviewer will obtain better information than that obtained using a well-developed and well-worded questionnaire.

Client Documents

Another way to obtain information about a client’s operations is to examine the forms that are currently used. E.g. A purchase order may carry fields such as stock no., product code, quantity, etc. which shed light on the work flow in the organization.

Other documents such as job descriptions, internal reports, operating procedures etc. can also be very helpful.

Observations

Use of video cameras can be used to monitor what actually happens on the job in the office environment. The presence of cameras in the workplace may represent a privacy issue so careful consideration of all factors and the risks involved should be taken before embarking on this path. Legal issues may also be involved.

Scenarios

A scenario is a possible way in which a user can utilize the target product to accomplish some specific objective.

The developer may give a scenario of what output will be given by the product-to-be for a given input. The client representative then indicates what modifications will be needed for the scenario to correspond to what happens in practice.

Scenarios attempt to illustrate the behavior of a future product in a way that the user can easily understand. This can result in new requirements being discovered by the developer.

Rapid Prototypes

A rapid prototype can be built to exhibit key functionality of the product-to-be. The client/users then will experiment with the prototype. The development team watches the experimentation and makes notes. Users can also indicate where they feel changes are necessary. The rapid prototype is changed several times until both developers and users are satisfied that the rapid prototype currently embodies the key needs of the clients.

The rapid prototype then becomes a basis for creating the specifications document.

*** *Key Point*** – Rapid prototype must be built quickly and be capable of easy on-the-spot modification. This means being written in a 4GL or interpreted language (Such as Prolog, lisp, or java)

2.3. Feasibility Study

A feasibility study is carried out to select the best system that meets performance requirements.

The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system.

Technical Feasibility

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably, but might include :

- The facility to produce outputs in a given time.
- Response time under certain conditions.
- Ability to process a certain volume of transaction at a particular speed.
- Facility to communicate data to distant locations.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should give the complete picture about the system's requirements:

How many workstations are required, how these units are interconnected so that they could operate and communicate smoothly.

What speeds of input and output should be achieved at particular quality of printing.

Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as Cost / Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an outgoing effort that improves in accuracy at each phase of the system life cycle.

Operational Feasibility

This is mainly related to human organizational and political aspects. The points to be considered are:

- What changes will be brought with the system?
- What organizational structure are disturbed?
- What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

This feasibility study is carried out by a small group of people who are familiar with information system technique and are skilled in system analysis and design process.

Proposed projects are beneficial only if they can be turned into information system that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed.

2.4. Algorithm

Algorithm is a step-by-step description of how to solve a particular problem. An algorithm provides step by step description of various methods to solve a problem. It is an effective procedure for solving a problem in a defined number of steps. Algorithm maintains sequences of computer instructions required to solve a problem in such a way that if the instructions are executed in specified sequence.

The Desirable Features of an Algorithm are:

1. Each step of the algorithm should be simple.
2. It should be unambiguous in the sense that the logic should be clear.
3. It should be effective.
4. It must end in finite number of steps.
5. It should be an efficient as possible.
6. One or more instructions should not be repeated infinitely.
7. Desirable result must be obtained on the algorithm termination.

Example for developing an algorithm is steps of program design. Let us consider an example of an algorithm for making a tea.

Step 1: Start

Step2: Put water in Kettle

Step 3: Plug the Kettle into switch.

Step 4: If the water in the kettle is not boil, then go to step 3.

Step 5: Switch off the kettle

Step 6: Pour water from the kettle into the teapot.

Step 7: Stop

The algorithm shows the following three features:

Sequences (Process): Sequences means that each step or process in the algorithm is executed in the specified order. Each process must be in proper place previous steps must be executed before any other next steps.

Decision (Selection): In some cases we have to make decision to do something. If the output of the decision is true, one thing is done otherwise other control should execute. The outcomes of decision either true or false, there is not state in between them. For eg. If the number is less than zero, then the number is negative.

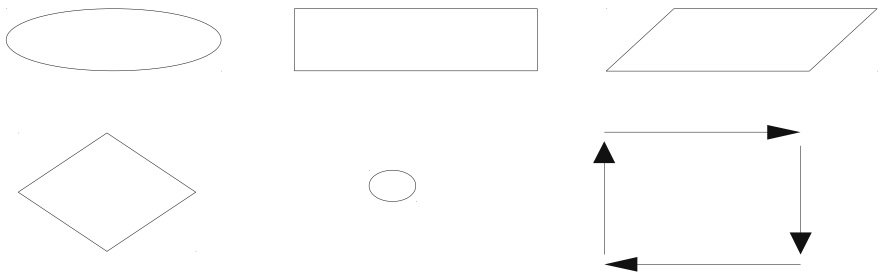
Repetition (Iteration or loop): Repetition can be implemented using control statements.

2.5. Flowchart

The flowchart is graphical representation of an algorithm using standard symbols. In other words, flowchart is a pictorial representation of an algorithm that used different geometric pictures for different instruction. The flowcharts play vital role in programming to solve problem and they are quite helpful in understanding the logic of complicated and lengthy problems. Once the flowchart is drawn it becomes easy to write the program in any high level language. Hence the flowchart is better documentation of the complex program. Flowchart is also a very effective analytical tool. With the help of a flowchart, programmer can quickly show a series of alternative approaches to solve problems.

Symbols Used in Flowchart

The various flowchart symbols suggested by ANSI are as follows:



Terminal Symbol: It is used to indicate a point at which the flowchart begins or ends. The words START & STOP are written within the terminal symbol. It is represented by geometric “oval” shape.

Processing Symbol: This symbol represents some operations on data. It is represented by geometric “rectangle” shape.

I/O Symbol: It is used to represent the logical positioning of input/output operation. It is represented by geometric “parallelogram” shape.

Decision Symbol: This symbol represents a logical operation showing a decision point in a program. It is represented by geometric “rhombus” shape. The two main components of a decision symbol are:

- A question that defines the logical operation.
- The result of the decision (yes, no)

Connector Symbol: It is used to indicate a junction at whom the flowchart comes from a part of the flowchart on another page. It is represented by geometric “circle” shape.

Flow Symbol: A flow symbol is an arrow that shows the flow of program logic in a flowchart. It is represented by “arrow headed line” shape.

Advantages

- **Better Communication:** Flowcharts are a better way of communications it quickly provides logic, ideas and detailed descriptions of computer operations.
- **Effective Analysis:** Flowchart provides a clear overview of the entire problem and its algorithm for solutions. IT shows all major elements and their relationship among components.
- **Proper Documentation:** The flowchart provides a permanent programming logic. It documents the steps followed in an algorithm. A clearly comprehensive flowchart is an indispensable part of documentations for each program.
- **Efficient Coding:** Flowcharts shows all major parts of a program. Programmer can easily instruct the computer in any platform. The flowchart specifies the steps to be coded and help to prevent errors. Thus flowchart is blue print of system analysis and program development phase.
- **Easy in Debugging:** Flowchart helps easy debugging and maintenance of operation in program.
- **Better Understanding:** A flowchart is a pictorial representation of a program. Hence it is easier for a programmer to explain the logic of a program through flowchart.
- **Easy to Convert:** Easy to convert flowchart to programming language .

Limitations

- **Complex Logic:** A flowchart becomes complex when the program logic is quite complicated.
- **Difficulty in alternation and modifications:** if alterations are required the flowchart may need to redrawn completely.
- Very time consuming and laborious job .
- Redrawing a flowchart is a tedious task .
- How much to include in flowchart is unclear.

2.6. Coding, Execution, Debugging and Testing

Coding

The coding is process of transforming program logic design into a computer language format. This stage translates program design into computer instructions using some programming language like C. The coding is the act of transforming operation into program statement. The knowledge of computer programming language is necessary to write coding. The code written using programming language is also known as source code. During coding of program, programmer should eliminate all syntax and format errors form the program and all logic errors are detected and resolved during process.

Compilation and Execution

The process of changing high level language into machine level language is known as compilation. It is done by special software, known as compiler. Compilation process tests the program whether it contains errors or not. If syntax errors are present, compiler cannot compile the code. Once compilation is completed then the program is linked with other object program needed for execution. Thereby resulting in binary program and then program is loaded in the memory for the purpose of execution. During execution program may ask user for inputs and generates outputs after processing the inputs.

Debugging and Testing

Debugging is the recovery and correction of programming errors. Even after taking full care in program design and coding, some errors may remain in the program become the programmer might never about case. These errors may appear during compilation or execution of program. When the errors are appeared the debug is necessary. Testing ensure that program perform correctly the required task. Thus programming testing and debugging are very closer.

Program Documentation

Program documentation is description of program and its logic written to support understanding the program structure. Documentation of program helps to use and extend the program future. A program may be difficult to understand even to the programmer who wrote the code after same day. If a program coded by one person is to maintained, there will be more difficult to understand it. Proper documentation is necessary which will be useful and efficient in debugging, testing, maintained and redesign process.

Historical Development of C Language

During 60's there were a number of programming languages developed but almost all were used for specific purpose only. For example FORTAN (Formula Translation) developed by IBM was used for engineering and scientific applications. COBOL (Common Business Oriented Language) developed in 1959 was used for commercial applications especially for business purpose. Due to those specific purpose languages the programmer had to learn more. i.e. one language was not for sufficient for every field.

Therefore computer scientist started to think for common language for all possible applications. As a result (ALGOL) Algorithm Language was developed by European and American, but ALGOL never really become popular because it seems to general purpose programming language only.

To eliminate this problem CPL (Combined Programming language) was developed by the mathematical Laboratory at Cambridge University of London. This collaborative effort was responsible for combined in the name of the language. It was heavily influence by ALGOL. The new language CPL is developed with mixture of FORTAN and COBOL but this is not become more popular.

The next efforts the Basic CPL is designed by Martin Richard of the University of Cambridge in 1966, was much simpler language primarily as a system programming language, particularly for writing compiler. At the same time the language B was developed at Bell Labs. It was mostly the work of Ken Thompson with contributions for Dennis Ritchie in 1969. But like BCPL was not become much popular one. At 1972 C was written and designed by Dennis Ritchie at AT and Bell Lab for the use of UNIX operating system by inheriting the features of B and BCP language add more features. The origin of C is closely to the development of UNIX operating system. It was named "C" because many of its features were derived from an earlier language "B". C is a general purpose, structure, procedural imperative language. Some of the most common C compilers are Turbo C/C++ IDE, Borland C/C++, and Microsoft visual C++ etc.

Importance/Advantages of C Programming Language

Robust Language: C is robust language because which rich set is of built in functions and operators can be used to write any complex program. C compiler combines the capabilities of an Assembly language with the features of high level language.

Efficient and Fast: Programs written in C are efficient and fast due to its variety of data types and powerful operators.

Highly Portable: C is highly portable; meaning that C program written for one computer can run on another computer with little or no modifications of source code.

Structure Language: C is structured language as it has a fixed structure. C program can be divided into numbers of block or modules. Thus, the proper collection of modules would make complete program.

Extensible: C programs may contain a number of user defined functions. We can add our own user defined functions to the C library if required.

Middle Level Language: C is middle level language because it combines the best part of high level language with low language. It is well suited for writing both user and machine oriented program.

Rich System Library: There are large number of built in functions, keywords and operators in C system library supports.

Execution of C Program

1. **Writing the Source Code**

   Computer instructions are written in a text editor to perform certain jobs. These instructions written using the syntax of C is known as the source code of C program. The source code can be written using any text editor such as Notepad, Turbo C. However it should be saved with ".C" extension. For e.g. "first.c" is valid file name.

2. **Compiling and Linking the Program**

   The computer instructions written in the form of source code are translated into executable code with the help of compiler that is suitable for program execution. The translation is done by a special program called compiler that processor statements written in programming language and converts them into a machine level language. So compiling means creating an executable file for particular platform. The compiler first analyzes the entire language statements syntactically one after another and then build the output code called object code. The object codes are machine code that the processors can process or execute one instruction at a time.

   During compilation linking process takes place. Linking is the process of putting translated program and other objects from system library such as reading inputs, producing output and computing mathematical functions. To create an executable program the object program must be linked to system library subprogram.

3. **Execution of Program**

   While execution of program the loader loads executable object code into the computer memory that executes the instruction. During execution, the program may require for some data to be entered through the keyboard.

Basic Structure of C Program

The structure of C program implies composition of a program it describes main components to write in C program, how are they organized. The following table shows the parts are included in the structure of C program.

- Documentation Section
- Linking Section
- Definition Section
- Global Declaration
- main() function
- Sub program()

1. Documentation Section

This section contains a set of comments lines giving the name of program, the designer may write algorithm, methods used and other detail information related to the program. This will be useful in further for users and development teams. Documentation acts as a communication medium between members of development team working in the same project. It helps while debugging and testing the program.

```
/* This program display natural numbers from 1 to 10 */
```

Note: /----- */ denotes comments in C Whatever the text written within comment if just ignore by the compiler.*

2. Linking Section

This section provides instruction to the compiler to link functions with program from the system library.

```
#include<stdio.h>

#include<conio.h>
```

It links system function library I/O that handles printf() and scanf()functions. They are preprocessors which provide the designer platform to design program.

3. Definition Sections

In this section all symbolic constants are defined.

```
#define PI 3.1416
```

4. Global Declaration

The variables which are used in more than one function/block are called global variables. Those variables are defined in this section. This section also defined all the user defined functions.

5. Main Functions

Every C program must have one main function through which program starts execution. The main function has deceleration and execution parts. Deceleration part declares all the variables used in the program execution part. In this section some sort of calculation and other functionality are to be kept. e.g.

```
int i,k;              /*deceleration of integer variable to store integer value*/
i=90,k=4*i;           /*execution part and assignment parts.*/
```

6. Subprogram

This section contains all the user defined functions that are called in the main function.

Example C Program

```
/*C program display "Hello World"*/
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Hello World.");
        getch();
}
```