UNIT 1 SYSTEM DEVELOPMENT

Structure

1.0 Introduction
1.1 Objectives
1.2 Tasks of System Development
1.3 Prototype Installation
1.4 Hardware and Software Selection and Performance
  1.4.1 Hardware Selection
  1.4.2 Software Selection
1.5 Benchmark Testing
1.6 Preparing Software Development Cycle
  1.6.1 Identifying Programs
  1.6.2 Program Logic and Flowcharts
  1.6.3 Control Structure
  1.6.4 Pseudocode
1.7 Software Specification Language Selection Criteria
  1.7.1 Volume of Data
  1.7.2 Complexity of Processing
  1.7.3 Compatibility with other Systems
  1.7.4 Types of Input/Output
  1.7.5 Development Efforts
1.8 Summary
1.9 Model Answers

1.0 INTRODUCTION

The fourth phase in the life cycle of a system is the software development. In this development phase, the computer- based business system is developed to conform to the design specification prepared in the preceding phase. This phase involves heavy expenditure because of recruiting additional staff for the purpose of software development, purchase of machine, materials and the use of computer facilities. The principal activities performed during the development phase are: (a) External system development and (b) Internal system development. The major activities that comes under external system development are implementation planning; preparation of manuals and personnel in-house training; and equipment acquisition and installation. Software development and performance testing are considered to be the principal activities of internal system development.

1.1 OBJECTIVES

After completing this unit, you should be able to:

- understand different tasks of system development
- know the importance of prototype
- find out various factors to be considered prior to system selection
- understand the term 'benchmark'
- define the various parameters responsible in considering the selection of a language

1.2 TASKS OF SYSTEM DEVELOPMENT

Major tasks of system development have been summarised below:

(i) Implementation Planning: After the initiation of development phase is approved, implementation planning starts. Essential parts of implementation plan are:

(a) A plan for testing the computer program component, both as the integrated assembly of its individual programs and as an element of the overall business system.

(b) A plan for training the personnel associated with the development of software on the new system. This includes persons who will provide inputs to, receive outputs from, operate or maintain the new system.

(c) A conversion plan that provides for the conversion of procedures, programs and files preparatory to actual changeover from the old system to the new one.

(ii) Software development phase: Software development phase can work along with the implementation planning efforts. If it is necessary, system flowcharts are expanded to show additional detail for the computer program components. The complete database is developed. Input and output files are identified and computer program logic flowcharts are prepared for each computer program component.

(iii) User Review: Reviews are held with the principal user throughout the development phase. A review of test plans, training plans, and conversion plan is quite important because users are directly involved in implementation activities. Users' concurrence with the implementation plan is extremely important to carry on the software development work in an efficient way.

(iv) Equipment acquisition and installation: In the design phase, special hardware required to support the system may have identified. If the hardware is not ordered during the design phase, it is proper time to go for it. It is also true that all hardware components are not required at a particular time because the needs vary depending upon the type of software being developed. It is, therefore, necessary that a proper schedule should be prepared for acquisition of hardware components.

(v) Coding, debugging and testing of computer program: Each of the computer programs that make up the entire system is coded and debugged. This means that each computer program is compiled error free and successfully executed using the test data prepared by the programmer.

(vi) System testing: System tests are performed to verify that the computer based business system has met its design objectives.

(vii) Reference manual preparation: Proper reference manual for the various individuals who will be associated with the new computer based information system must bc prepared.

(viii) Personnel training: Operating, programming and user personnel are trained using the reference manuals, forms and procedures as training aids. All sort of training activities must be completed prior to the user acceptance review which occurs at the end of the development phase.

(ix) User acceptance review: At the end of development phase, the computer based system is reviewed by the management of the user organisation. Representatives of the information service organisation and other affected organisations take part in this review. Design phase report, development phase report and test reports are some of the important documents which are responsible for the acceptance review.

1.3 PROTOTYPE INSTALLATION

A prototype is the process of creating, developing and refining a working model of a final system. It does not contain **all** the features or perform all the necessary functions of the final system. Rather, it includes large number of elements to enable individuals to use the ***proposed*** system to determine what they like and do not like and to identify features to be added or changed. Application prototyping, the process of developing and using the prototype, has the following characteristics:

(i) The prototype is a live, working application.

(ii) Its main purpose is to test out the assumptions made by the analysts and users about the features of required system.

(iii) Prototypes can be quickly created.

(iv) They follow an iterative process.

(v) They are relatively cheap.

Application prototyping has two primary uses. The first one is that it is an effective device for clarifying user requirements. Written specifications are typically created as a vehicle for defining application features and the requirements that must be satisfied.

A second use of application prototyping is to verify the feasibility of a system design. Analysts can make experiment using different application characteristics, evaluating user reaction and response.

The rationale for application prototyping is a direct outgrowth of the need to design and develop information systems quickly, efficiently and effectively.

Application prototyping is a proven technique that improves the overall effectiveness of the development effort for the benefit of user, analyst and the organisation as a whole.

1.4 HARDWARE AND SOFTWARE SELECTION AND PERFORMANCE

1.4.1 Hardware Selection

The decision to acquire computer hardware or software must be handled in the same way as any other business decision. The variety of sizes and types of computing resources available puts a burden on the analyst who must select suitable hardware, software or services and advise the top management accordingly.

Today, selecting **a** system is a serious and time-consuming business. The time spent on the selection process is a function of the applications and whether the system is a basic micro-computer or a mainframe. In either case, planning system selection and acquiring experienced help where necessary pay off in the long run.

There are various important factors which should be considered prior to system selection. They are:

(a) Define system capabilities that make sense for the business.

(b) Specify the magnitude of the problem; that is, clarify whether selection entails a few peripherals or a major decision concerning the mainframe.

(c) Assess the competence of the in-house staff.

(d) Hardware and software should be considered as a package.

(e) Develop a time frame for the selection process.

(f) Provide user indoctrination. This is crucial, especially for first-time users. Selling the system to the user staff, provide adequate training and creating an environment conductive to implementation are pre-requisites for system acquisition.

The selection process should be viewed as a project and a project team should be formed with the help of management. The selection process consists of several steps which are discussed below:

1. Requirements analysis: The first step in selection is understanding the user's requirements within the framework of the organisation's objectives and the environment in which the system is being installed.

2. System specifications: System specifications must be clearly defined. These specifications must reflect the actual applications to be handled by the system and include system objectives, flowcharts, input-output requirements, file structure and cost.

3. Request **for** proposal: After the requirement analysis and system specifications have been defined, a request for proposal is prepared and sent to selected vendors for bidding.

4. **Evaluation and** validation: The evaluation phase ranks various vendor proposals and determines the one best suited to the user's requirements. It looks into items such as price, availability and technical support. System validation ensures that the vendor can, in fact, match his/her claims, especially system performance.

5. Vendor selection: This step determines the vendor with the best combination of reputation, reliability, service record, training, delivery time, lease/finance terms. The selected vendors are invited to give a presentation of their system. The system chosen goes through contract negotiations before implementation.

1.4.2 Software Selection

Software selection is a critical aspect of system development. There are two ways of acquiring software: custom- made or "off-the-shelf' package. Today, there is great demand for these packages because they are quite cheap. There are other benefits also.

(i) A good package can get the system running quickly.

(ii) MIS personnel are released for other projects.

(iii) 'Home-grown' software can **take** more time and its cost cannot be predicted.

(iv) Package can be tested before purchasing it.

Some drawbacks of software packages are:

(i) These packages may not meet user requirements in all respect.

(ii) Extensive modifications of a package usually results in loss of the vendor's support.

It can be observed that price alone cannot determine the quality of software. A systematic review is crucial for selecting the desired software. Prior to selecting the software, the project team must set up criteria for selection. The criteria for software selection are:

(a) Reliability: gives consistent results without any failure for a specified time period.

(b) Functionality: functions to standards.

(c) Capacity: satisfies volume requirements of the user.

(d) Flexibility: adapts to the changing needs

(e) **Usability:** is user-friendly.

(f) Security: maintains integrity and prevents unauthorised user.

(g) Performance: delivers the results as expected.

(h) Serviceability: has good documentation and vendor support.

(i) Ownership: has right to modify and share use of package.

(j) Minimal costs: is justified and affordable for intended application.

Check Your Progress 1

1. List out the major tasks of system development.

....

....

....

....

....

2. What arc the important factors to be considered prior to system selection?

....

....

....

....

....

3. Explain briefly about the criteria for software selection.

....

....

....

....

....

1.5 BENCHMARK TESTING

The term "benchmark" was derived from the days when the machinist in a factory would use measurements at each bench to determine if the parts he was machining were satisfactory. In the computing field, to compare one system with another, you would run the same set of "benchmark" programs, through each system.

A benchmark is a sample program specially designed to evaluate the performance of different computers and their software. This is necessary because computers will not generally use the same instructions, words of memory or machine cycle to solve particular problem. As regard, evaluation of software, benchmarking is mainly concerned with validation of vendor's claims in respect of following points:

— minimum hardware configuration needed to operate a package.

— time required to execute a program in an ideal environment and how the performance of own package and that of other programs under execution is affected, when running in a multi-programming mode.

The more elaborate the benchmarking, the more costly is the evaluation. The user's goals must be kept in mind. Time constraints also limit how thorough the testing process can be. There must be a compromise on how much to test while still ensuring that the software (or hardware) meets its functional criteria.

Benchmarks can be run in almost all type of systems environment including batch and on-line jobs streams and with the users linked to the system directly or through telecommunications methods.

Common benchmarks test the speed of the central processor, with typical instructions executed in a set of programs, as well as multiple streams of jobs in a multiprogramming environment. The same benchmark run on several different computers will make apparent any speed and performance differences attributable to the central processor.

Benchmarks can also be centered around an expected language mix for the programs that will be run, a mix of different set of programs and applications having widely varying input and output volumes and requirements. The 'response time for sending and receiving data from terminals is an additional benchmark for the comparison of systems.

Benchmark is one of the evaluation techniques used by the computer purchasers to determine which marking is best for them in marking out their requirements in terms of both speed and cost.

1.6 PREPARING SOFTWARE DEVELOPMENT CYCLE

Software development, which involves writing of programs, begins after systems design has been completed. Again, the work done in the previous phase is the foundation on which the work in this phase will be built, As long as system design has followed the principles of structured design, software development phase will start properly. But if the structured principle have not been properly followed, die results may be disastrous.

In the structure chart (Fig. 1), it is quite clear that an information system can be decomposed into a group of related modules, each of which reprecents a self-sufficient function within the new system. This modular approach, together with certain programming guidelines and regular reviews, represent the fundamental working concepts of structured programming.

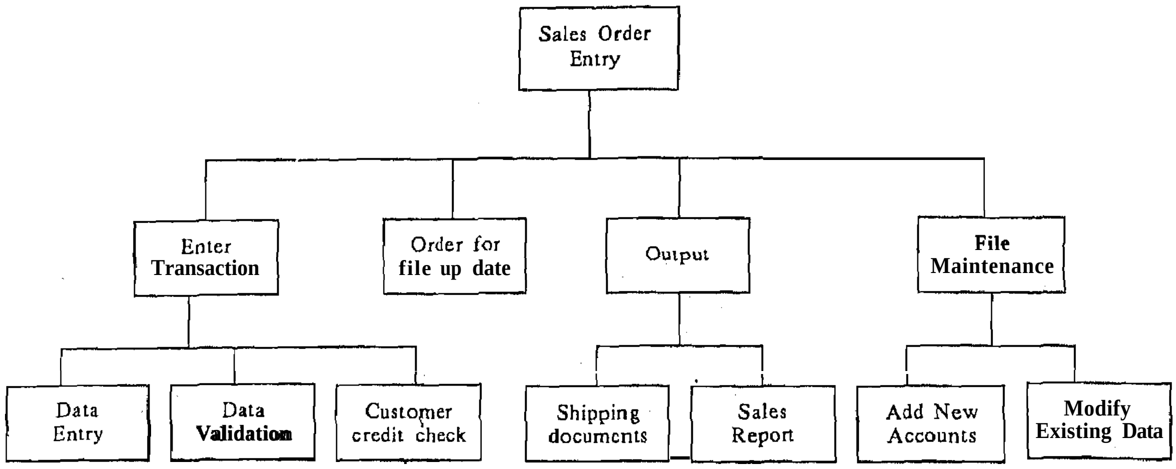


Fig. 1: A Structure Chart of a Sales Order Entry System

1.6.1 Identifying Programs

Every large system consists of number of small programs because these smaller programs are easier to write, modify, troubleshoot and maintain as compared to very large and cbmplex program. The modular approach serves as the basis for identifying separate programs.

1.6.2 Program Logic and Flowcharts

As we move closer to the time when programmers will start writing programs called coding. The detailed logic behind each program is generally the program flowcharts, data flow diagrams or in a more English-like form called pseudocode.

The program flowchart is a detailed graphical representation of the logical flow of data within that module. It serves as the logical road map that the programmers will use to write programming code. When following structured programming principles, the program flowchart must use the standard symbols as shown in Fig. 2 and certain guidelines concerning control structures which will now be described:

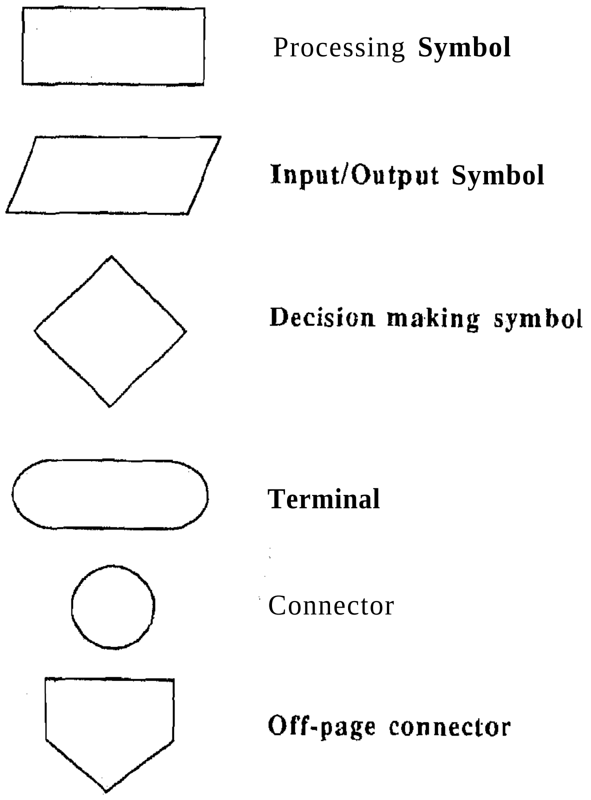


Fig. 2: Standard Symbols Used to Construct a Program flowchart

1.6.3 Control Structures

Any program whether it may be simple or complex, can be developed using only *three* basic control structures: (i) simple sequence (ii) if-then and (iii) do-while. Let us describe each briefly.

Simple Sequence: A simple sequence structure is used when a series of steps must be carried out in linear sequence. These steps begin with the first and end with the last, Fig. 3 shows a simple sequence containing some of the steps necessary to validate employee wage data, one module in a payroll system.

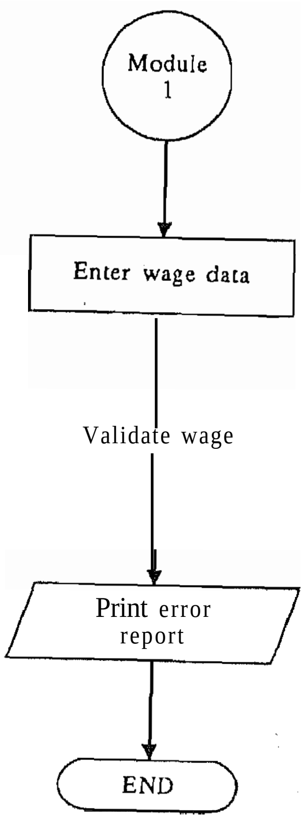


Fig. 3: A Simple Sequence Structure Example

If-then Structure: The if-then structure is used to transfer control from one point in a program to another. This transfer is based on meeting out certain condition. Suppose we want to write a program which all employees with more than 10 years of experience are to be listed in a specific report. Fig. 4 shows how the if-then structure can be used to perform this test

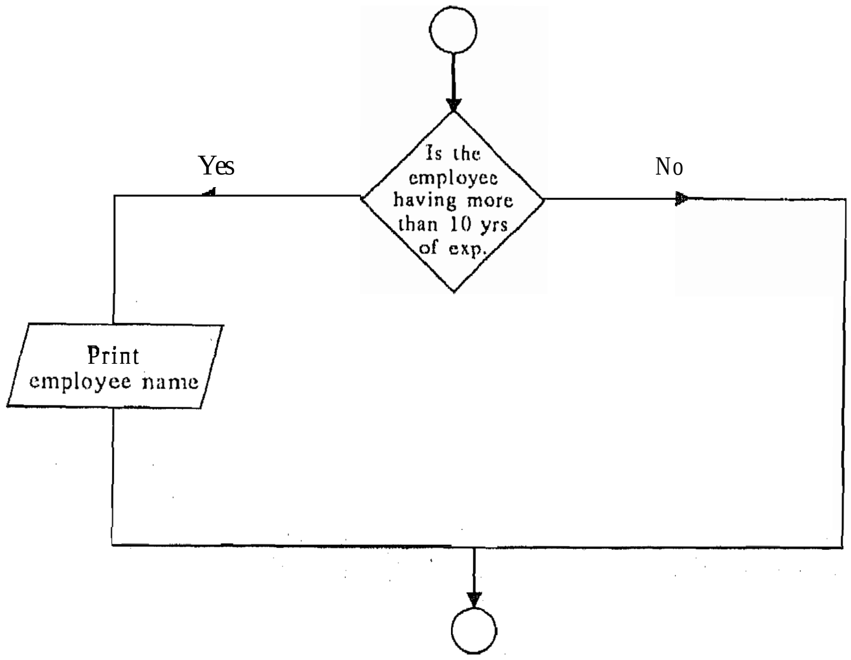


Figure 4: If-then Structure Example

Do-while **Structure:** The do-while structure is used when it becomes necessary to loop through or repeat over and over a sequence of steps. Looping starts at the entrance of the structure and continues while a pre-specified condition still exists.

It is important to remember that a do-while structure tests immediately for a pre-specified condition and does not allow any processing between the entry of the do-while structure and this test; Processing within the structure can only be done after the test has been made and in the line that returns to the top of the do-while structure. Therefore, a record must be read before a do-while structure is entered for the fist time.

Suppose we want to access an employee file, read each record and print its contents. As you can see from Fig. 5 a record is read before the do-while structure is entered for the first time. Immediately **upon** entering, the test for more record is made and If here are more record then the contents of the first record will be printed and another record will be read. This process continues until the last record. At that time, control passes out of the structure and to the next program step.

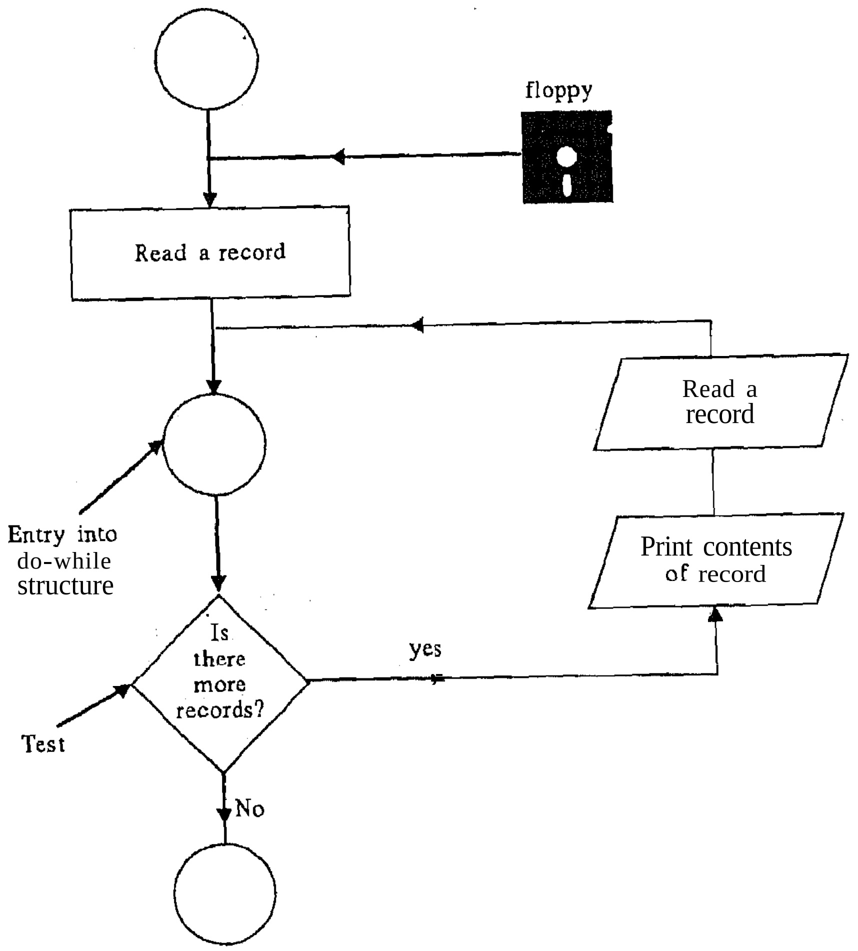


Fig. 5: A Do-while Structure Example

Combining **Control Structure:** Each of these three control structures can be combined to form complex flowcharts and complete programs. Fig. 6 is a complete program flowchart for a process that reads accounts receivable records from a file and prints a list of all those customers whose accounts are more than 60 days overdue and whose balance is greater than Rs. 1000.00.

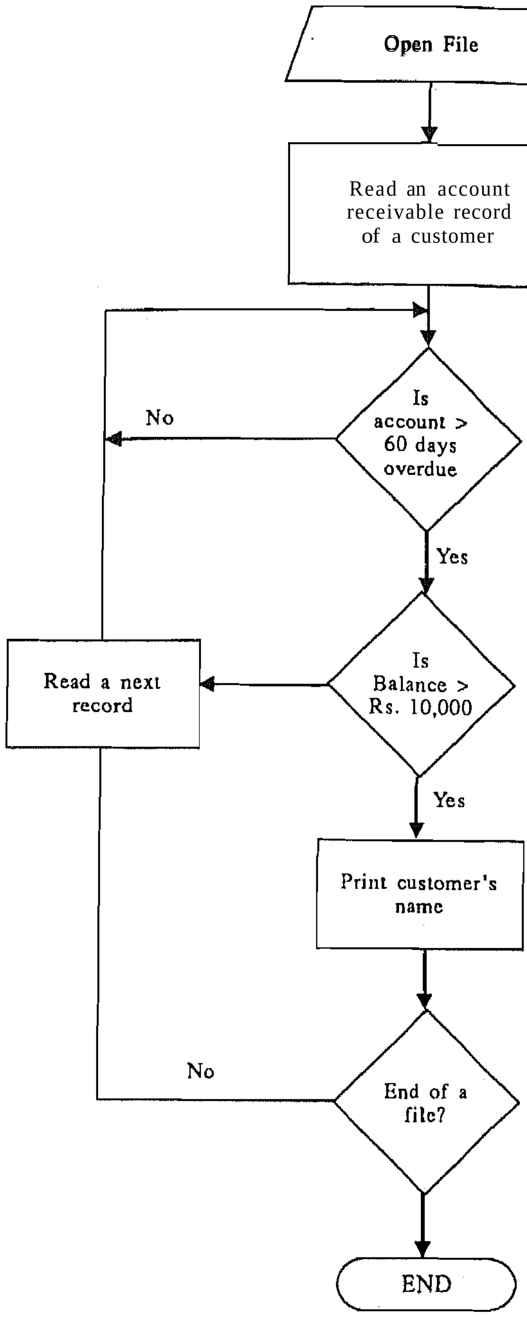
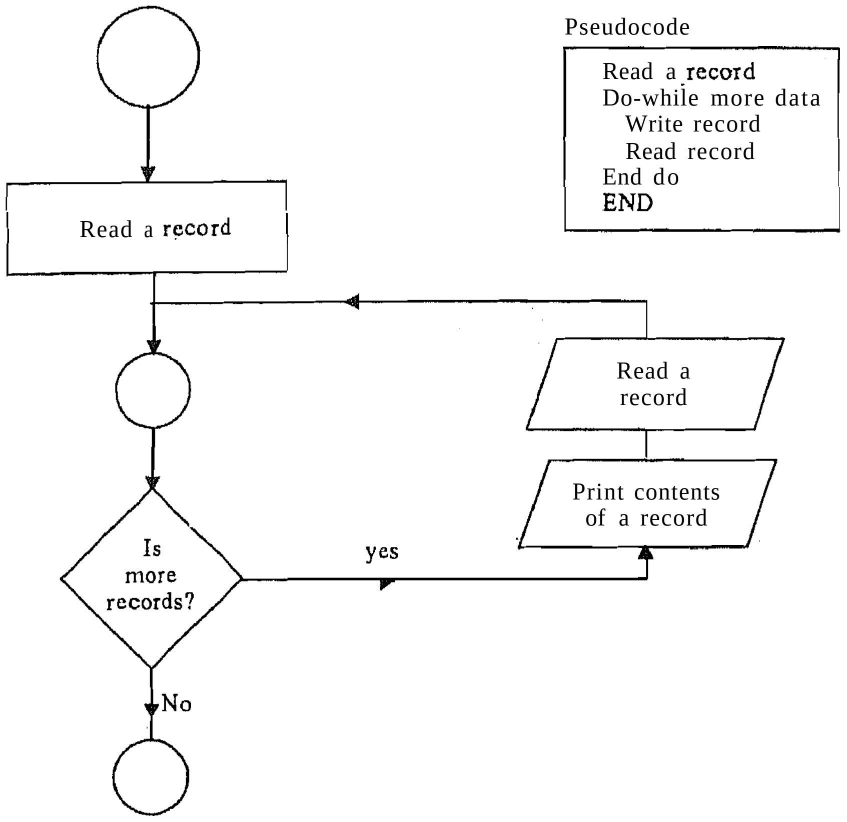


Fig. 6: A Complete Program Flowchart

1.6.4 Pseudocode

These program flowcharts show the logic of a program. Pseudocode often used as **an** alternative to this technique, expresses the logic of **a** program in English statements. It is a verbal rather than a graphic in nature. Fig. 7 shows various steps necessary to read **and** print the contents of a file with the help of flowchart and pseudocode.

One advantage of pseudocode is that it closely resembles **the** form that the actual programming code will take. Another advantage is that it avoids laying out symbols on paper.



```
Read a record
Do-while more data
   Write record
   Read record
End do
END
```

Fig. 7: Pseudocode and a Flowchart Equivalent

1.7 SOFTWARE SPECIFICATION LANGUAGE SELECTION CRITERIA

After the program flowcharts or pseudocode have been completed, a programming language must be chosen. Often that choice will depend on the language in which most of an organisation's other programs are written. This ensures consistency and eases the maintenance problem. Nevertheless, a number of different languages are currently in use, only a few of them need to be considered for the purpose of appreciating the issues,involved in choosing a programming language. The languages that are discussed are: FORTRAN, COBOL, BASIC, C and dBASE.

FORTRAN (FORmula TRANslation) is a compact language that serves the needs of both the Scientists and Economists. Main advantage of this language is that it supplies large library of Mathematical and Engineering sub-routines being utilised by the programmer in solving different type of problems of numerical and scientific in nature. Handling files containing voluminous data is definitely not,a strength of this language.

COBOL(COmmon Business Oriented Language) is one of the most popular languages used for large data processing problems. Its main strengths lie in handling files of large size and the ease of understanding and editing the program. COBOL can solve simple arithmetic problems but does not help in solving any complex mathematical problems.

BASIC (Beginners' All-purpose Symbolic Insrruction Code) is the most popular conversational programming language. It is simple to understand. Various versions of BASIC have been developed by computer manufacturers for their computers. It is suitable for both mathematical and business problems. It has been specially designed for use in time-sharing environment but can also be used as a standard programming language in a batch-processing environment, The main problem in BASIC language is that most versions of this language do not support indexed files.

C is a quite powerful programming language which is as compact as its name. C can be used where programmer wants to have more control over the hardware because it provides features that would typically be possible with machine/assembly languages only. This language is quite popular among people who are engaged in developing system software like operating systems or other utilities. The disadvantage of this language is that the program is not too easily decipherable and it takes a long time to learn.

dBASE is a Fourth Generation Language which is more an application development tool rather than a programming language. It provides features to store and retrieve data. The major advantage of this language is the query and reporting facility which helps in generating the report quickly. It is quite simple and easy to understand. If the volume of data to be processed is very large, the program can become quite slow.

Numbers of parameters responsible in considering the selection of a language are:

- volume of data
- complexity of processing
- compatibility with other systems
- types of input/output
- development efforts

1.7.1 Volume of Data

This covers two aspects:

(i) Number of files

(ii) Number of records

Some languages put a restriction on the number of files that can be accessed at a time. For example, dBASE III + can open ten data files only whereas COBOL does not have such restriction. It is also observed that most languages tend to become slower when the file size become larger. This slowing down feature due to large number of records is usually found in Fourth Generation Languages. There is also the other extreme where certain tools are not desirable for a very low volume of data. This is typically true of DBMS based products.

1.7.2 Complexity of Processing

Some applications take little bit input, do small calculation and generate output. There are quite simple reporting programs. These programs would not require much coinputation work. On the other hand, some applications involve plenty of computation, for example analysing data from a drilling rig exploring for oil. These would also benefit from the use of mathematical co-processors. FORTRAN is a language that is desired for such type of applications. Some spreadsheets also help in doing complex computations. In case the application does not involve complex computation but require repetitive computations, a language like COBOL or dBASE III would be desirable.

1.7.3 Compatibility with other Systems

Any new system cannot exist in isolation but has to co-exist with other systems. Generally we face the problem that some data files may have to be accessed by the old and new systems. If this is not adhered to, the result could be chaotic and may also require additional overheads. It is also possible that the two systems, if in different languages, may not be able to read the same data files e.g. COBOL programs cannot directly read a dBASE III file. This compatibility issue has to be kept in mind even when two parts of the system are written in different languages. In such a situation, it may not suffice that sharing of data is permitted, and two programs may need to invoke each other.

1.7.4 Types of Input/Output

Two issues involved me:

(i) Types of Input/Output device

(ii) Complexity of Format

While considering the types of input/output devices that are required, the answer is normally yes or no. For example, if a program needs to give output to a plotter, dBASE may not be suitable.

Complexity of format is little bit more difficult to decide. Some applications may definitely need graphic output, in which case the options are to use an integrated package like LOTUS 1-2- 3 or VP Planner, which have the capability of giving graphic output. There may be some applications which require complex formatted screens or printouts. In such cases, one method to be explored is "How critical is the format? Can it bc modified to make programming efforts simpler?"

For example, the user may have asked page number at the bottom of the report whereas the REPORT FORM of dBASE would print the page number on top of the page. Printing the page number at the bottom of the page may mean program writing instead of using the tools available. If the user wants adhoc queries then a powerful query language like dBASE III would be suitable.

1.7.5 Development Effort

The parameter could sometimes be the decider. The main reasons for different languages taking different amounts of effort for the same task are:

(i) features available in the language

(ii) learning time

A simple example of feature availability is that most versions of BASIC do not support indexed files whereas COBOL supports indexed files. The learning time required to gain proficiency could become a critical factor if software has to be developed in a language that the development taken is not familiar with. It is possible that even a particular language offers many good features but might take enough time to learn. It is very possible that programming language is chosen based on whal language the people in the organisation are familiar with.

Check Your Progress 2

1. Describe "Benchmark Testing" briefly.

   ....

   ....

   ....

   ....

   ....

2. List out various parameters responsible in considering the selection of suitable language in a big organisation.

   ....

   ....

   ....

   ....

   ....

1.8 SUMMARY

System analyst play very important role in developing information systems that are useful to management and employees in business systems. The systems development life cycle, the set of activities that analysts and designers carry out to develop and implement an information system.

Prototyping is an appropriate development strategy when predicting the user's requirement is not possible. A prototype, a version of an information system having the essential features but not necessarily all details of the user interface or 'performance efficiency, is developed and put into use.

In hardware/software selection also, system analyst should-act very carefully. He must consider all important factors prior to system selection.

Systems analysts rely on a wide variety of tools to fulfill their responsibilities. These tools can be very helpful in the process of system development and implementation.

1.9 MODEL ANSWERS

Check Your Progress 1

1. Major tasks of system development are:

   (a) Implementation planning

   (b) Software development phase

   (c) User review

   (d) Equipment acquisition and installation

   (e) Coding, debugging and testing of computer program

   (f) System testing

   (g) Reference manual preparation and training

   (h) User acceptance review

2. Important factors to be considered are:

   (a) Define system capabilities

   (b) Specify the magnitude of the problem

   (c) Assess the competence of the in-house staff

   (d) Consider hardware and software as a package

   (e) Develop a time frame for selection process

   (f) Provide user indoctrination

3. Criteria for software selection are:

(a) Reliability,	(b) Functionality	(c) Capacity	(d) Flexibility
(e) Usability	(f) Security	(g) Performance	(h) Serviceability
(i) Ownership	(j) Minimal costs		

Check Your Progress 2

1. A benchmark is a simple program specially designed to evaluate the performance of different computers and their software. This is necessary because computers often do not use the same instructions, words of memory or machine cycle to solve a particular problem.

2. Various parameters responsible in considering the selection of suitable language are:

   (a) Volume of data

   (b) Complexity of processing

   (c) Comparability with other systems

   (d) Types of input/output

   (e) Development efforts