System Analysis & Design

MCA

First Year
DAA-13

Manonmaniam Sundaranar University
Directorate of Distance and Continuing Education

Author: Preeti Mittal

Copyright © 2011, Excel Books Pvt. Ltd.
All Rights Reserved

Produced and Printed
by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi - 110 028
for
MANONMANIAM SUNDARANAR UNIVERSITY
DIRECTORATE OF DISTANCE AND CONTINUING EDUCATION
Abishekapatti - Tirunelveli 627 012, Tamil Nadu

CONTENTS

SYSTEM ANALYSIS & DESIGN

SYLLABUS

UNIT I

System Concepts and Information System Environment - Business System Concepts - Information - System Development life cycle - Introduction to CASE tools - Role of system Analysis - Communication skills.

UNIT II

Requirement Analysis and Methodologies - Sampling - Interviews - Questionnaires - Observing the Office Environment - Prototyping - Structured System Analysis Techniques and Practices - Cost Benefit Analysis.

UNIT III

System Design - The Process and Stages of System Design - Input/Output and Forms Design - File Organisation - Database Design.

UNIT IV

System Implementation - Software Maintenance - Review Plan - Hardware, Software Selection and the Computer Contract - Project Scheduling.

UNIT V

System Testing - Quality Assurance - Test Plan - Quality Assurance Goals - Audit Trail - Security, Disaster/Recovery and Ethics in System Development.

UNIT I

LESSON

1

SYSTEM CONCEPTS AND INFORMATION SYSTEM ENVIRONMENT

CONTENTS

1.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Define a system and a sub-system
- Differentiate between system approach and system analysis

A system is a set of interrelated elements that collectively work together to achieve some goal. For instance, accounting is a system with elements, viz., journals, ledgers, people, etc. and its basic goal is to maintain book of accounts alongwith preparation of financial and MIS statements. Computer is also a system with elements such as CPU (Central Processing Unit), input device, output device and users; and its basic goal is to process the data and provide information. There are hundreds of definitions of the word 'System', but here we define it as follows:

A system is a set of interrelated elements that form an activity or a processing procedure in order to achieve a common goal or goals by operating on data to yield information.

1.3.1 Subsystems

Most systems are part of a larger system. For instance, Financial Accounting System, Marketing System, and HRD (Human Resource Development) System are parts of a larger system, MIS (Management Information System) and are called subsystems. A system can be made up of many subsystems. A subsystem is defined as follows:

A subsystem is that part of a system that carries one part of the system function.

1.4 SYSTEM STUDY

Systems study may be defined as "a study of the operations of a set of connected elements and of the inter-connections between these elements". It shows clearly that one cannot ignore any part or element of a system without first finding out the effect that element has on the operation of the system as a whole. We can understand this with the help of systems analysis.

1.5 SYSTEM APPROACH

The formation systems (such as MIS) are designed on the basis of synergy of subsystems (such as Production, Inventory, Sales and Marketing systems) in order to achieve a net unified cohesive system.

The approach in developing information systems involves focus on the design of a whole integrated system rather than on independent subsystems in order to optimise the net results of the operations of an organisation. This is called the systems approach.

For instance, an invoicing system, an inventory control system and a financial accounting system can be designed independently. However, the net results of the operations of an integrated whole system are more than that of independent subsystems.

1.6 DIFFERENCE BETWEEN SYSTEM APPROACH AND SYSTEM ANALYSIS

There is a difference between "systems approach" and "systems analysis" also. The systems approach shows a set of procedure for solving a particular problem. It applies scientific methods to observe, clarify, identify and solve a problem with special care being taken to understand the inter-relatedness between elements and their system characteristics. However, systems analysis is a management technique which helps us in designing a new system or improving an existing system.

1.7 SYSTEM CHARACTERISTICS

A system has the following characteristics:

1. ***Organisation:*** Organization implies structure and order. It is the arrangement of components that helps to achieve objectives. The various elements of a system are organized to achieve objectives. For instance, input devices, output devices and the CPU of a computer system are organized to process the data and produce information.
2. ***Interaction:*** Interaction refers to the procedure in which each component functions with other components of the system. The various elements of a system are interacted with others to achieve a common goal. For instance, the ledger, journals and people are interacted in a financial accounting system for preparing the final financial statements (e.g., Profit and Loss A/c, Balance Sheet, etc.) of an organisation.
3. ***Interdependence:*** Interdependence means that components of the organisation or computer system depend on one another. The various subsystems of a system depend on one another for sharing of input data. For instance, in a computerised MIS (a system), the financial accounting system (a subsystem) receives the input data (e.g., financial data from Invoices, cash memo etc.) from the invoicing system (a subsystem).
4. ***Integration:*** Integration is concerned with how a system is tied together. It is more than sharing a physical part or location. It means that parts of the system work together within the system even though each part performs a unique function. Successful integration will typically produce a better result as a whole rather than if each component works independently.
5. ***Central objective:*** Central objective is the last characteristic of a system. Objectives may be real or stated. Although a stated objective may be the real objective, it is quite common that organisation may set one objective and operate to achieve another. The important point is that users must be aware about the central objective well in advance.

1.8 ELEMENTS OF SYSTEM ANALYSIS

There are four basic elements in systems analysis:

1. ***Outputs and Inputs:*** A major objective of a system is to produce an output that has value to its user. Whatever the nature of the output (goods, services or information), it must be in line with the expectations of the intended user. Inputs are the elements (material, human resources, information) that enter the system for processing. Output is the outcome of processing. A system feeds on input to produce output in much the same way in which a business brings in human, financial and material resources to produce goods and services.

   Defining aim is very vital in system work. If we do not know where we want to go, we will not know when we have reached there. We shall be unnecessarily wasting our time and energy in the process. Once we know our aim, we can try to achieve it in the best possible way. The user department has to define these objectives in terms of their needs. These becomes the outputs which the systems analyst keeps into mind.

   Once we know the output, we can easily determine what the inputs should be. Sometimes, it may happen that the required information may not be readily available in the proper form. This may be because the existing forms are not properly designed. If the information is vital to the system, we should make all possible efforts to make it available. The essential elements of inputs are:

(a) Data should be accurate. If data is not accurate, the outputs will be wrong.

(b) Data should be obtained in time. If data is not obtained in time, the entire system falls into arrears.

(c) The inputs must be available in proper format.

2. ***Files:*** Files are used to store data. Inputs necessary for the system are stored in files either in terms of isolated facts or in large volumes.

3. ***Processor(s):*** The processor is an element of a system that involves the actual transformation of input into output. It is operational component of a system. Processors may modify the input totally or partially, depending upon the specifications of the outputs.

4. ***Feedback:*** Control in a dynamic system is achieved by feedback. Feedback measures output against a standard in some form of cybernetics procedure that includes communication and control.

Feedback may be positive or negative, routine or informational. Positive feedback reinforces the performance of the system. It is routine in nature. Negative feedback generally provides the controller with information for action. In systems analysis, feedback is important in different ways. During analysis, the user may be told about the problems in a given application after receiving proper feedback. The user informs the analyst about the performance of the new installation. This feedback could be pointed or stored in an electronic format.

1.9 TYPES OF SYSTEM

Systems can be classified into various types as described below:

1. ***Physical and Abstract Systems:*** Physical systems are made of physical entities, viz., people, materials, machines, energy, etc. and intangible things viz., information, ideas, policies, procedures, etc. Physical systems are also called Empirical Systems. Some of the examples of physical systems are

   (a) Public transport system

   (b) Human digestive system

   (c) Computer hardware system

   (d) Computer software system

   (e) Management information system

   Abstract systems, on the other hand, are composed of only ideas, policies and theories and not of any physical entity. They are concerned with theoretical structures that may or may not exist in real life. They are also called conceptual systems. Some of the examples of abstract systems are -

   (a) A system of religious belief

   (b) A system of scientific theory

   (c) A system of economic theory

2. ***Open and Closed Systems:*** An open system interacts with its environment by receiving inputs from and sending outputs to outside the system. Almost all systems are open because they are affected by the environmental influences. A closed system does not interact with its environment. As it is isolated from environmental influences, it is not affected by changes occurred outside the system. In real life, closed systems are rare because almost all systems are affected by environmental changes. However, a laboratory system devised by scientists for research works can be considered a closed system, if it is isolated from its environment.

3. ***Social, Machine and People-Machine Systems:*** Systems composed only of people are called social systems. In reality, social systems are difficult to find because no system can achieve its goal without utilising some machines or equipments. However, a political party, a social club or a market association are generally considered as social systems though they may utilise some equipments. Contrary to social systems, machine systems are composed only of machines. Again in reality, machine systems are rare because machines obtain their inputs (data and instructions) from people and so depend on them. The computer-controlled, self-sufficient machines and Robots can be considered as machine systems. Systems composed both of people and machines are called People-Machine Systems.

4. ***Manual and Computer-based Information Systems:*** Before distinguishing between manual and computer-based information systems, let us first define an information system –

   An information system is a people-machine subsystem of the business system that supports the operational, managerial and decision-making information needs of an organisation.

   An information system can be either manual or computerised. An information system made of people, equipments, ideas, plans, procedures or other things (tangible or intangible) without using information or communication technology is called a manual information system. Although, computers are becoming a part of every information system, some organizations still use manual, payroll, accounting, inventory, invoicing and other business systems. A Computer-based Information System (CBIS) is an information system that consists of computer hardware, computer software, data, procedures and people as shown in Figure 1.1. A computerised financial accounting system, export documentation system, library management system and medical diagnostics system are some of the examples of computer-based information systems.

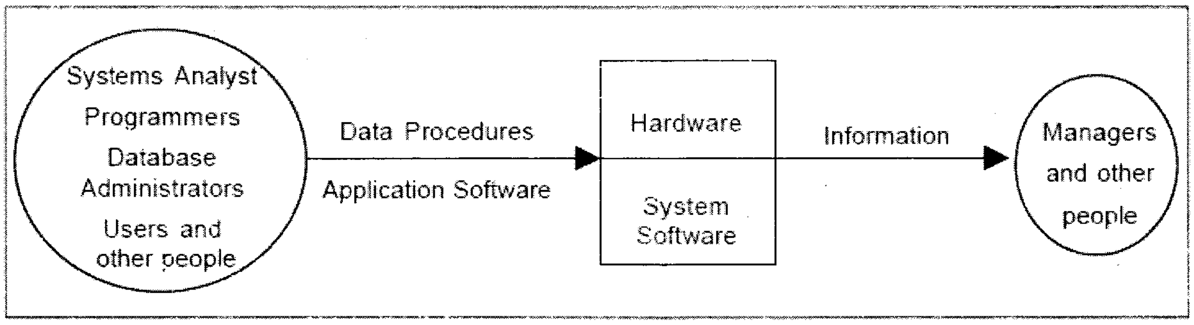


Figure 1.1: Basic Components of a Computer-based Information System

1.9.1 Types of Information System

There are five major types of information systems for various management levels of an organization which are illustrated in Figure 1.2 and are discussed below:

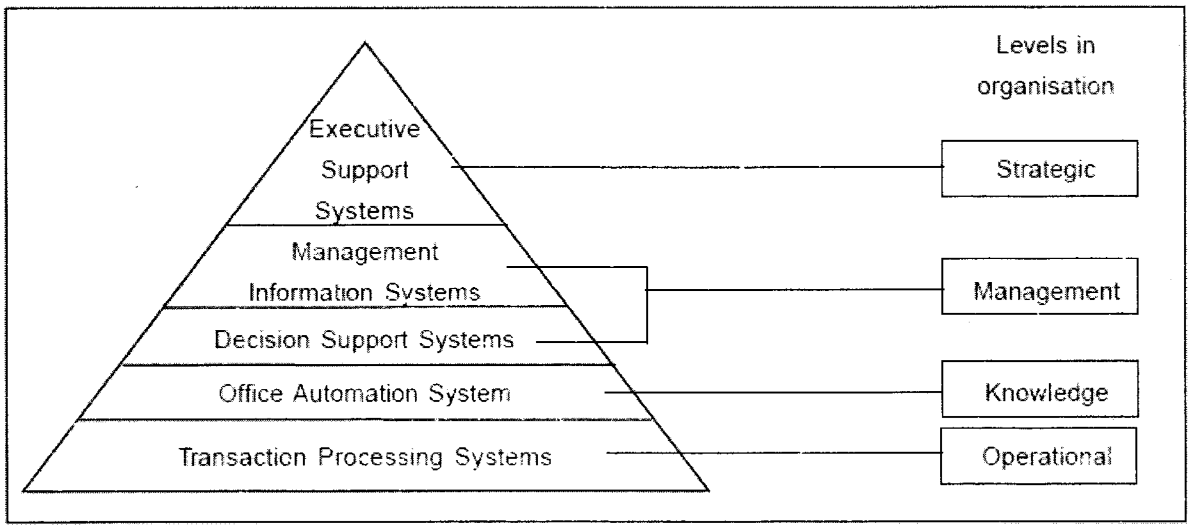


Figure 1.2: Types of Information Systems Supporting Different Levels of Management

1. ***Executive Support System (ESS):*** This system is designed to address unstructured decision-making at the strategic level of an organization. The systems at strategic level help senior managers in long-term planning. ESS employ advanced graphics and communications software for creating a generalized computing and communications environment.

2. ***Management Information System (MIS):*** This system is designed to serve the functions of planning, controlling and decision-making at the management level of an organization. The system at management level support monitoring, controlling and decision-making activities of middle level managers.

3. ***Decision Support System (DSS):*** This system also serves the information needs at management level of an organization. DSS differ from MIS in mainly having more analytical power and more user-friendly capabilities. DSS combine data and analytical/modeling tools to support semi-structured/unstructured decision- making.

4. ***Office Automation System (OAS):*** This system serves the knowledge level of an organization for supporting knowledge workers like production managers, EDP managers, etc. OAS use computer system to increase the productivity of technical managers in the office.

5. ***Transaction Processing System (TPS):*** This system is designed to serve the operational level of an organization. TPS record and process the daily routine transactions of the organization like, accounting, payroll, order processing, etc.

6. ***Stationary and Non-stationary System:*** The operations and properties of a stationary system do not change significantly while those of a non-stationary system change with time. For example, a computerized MIS is a stationary system because once designed, the MIS handles problems and provides information on a routine basis without any significant changes. An organizational system that tends to adapt to a changing environment is an example of a non-stationary system.

7. ***Adaptive and Non-adaptive System:*** Adaptive System tend to adapt to a changing environment while non-adaptive systems do not adapt. For example, an organizational system is an adaptive system and a MIS system is a non-adaptive system. Stationary system is always non-adaptive while non-stationary systems are adaptive systems.

1.10 COMPONENTS OF SYSTEM

A system is composed of various components which can be tangible or intangible objects. A tangible object is one that can be touched or measured. For example, computers, car, people, equipments, etc., are tangible objects. An intangible object is one that cannot be touched or measured. It can be an abstract concept (e.g., data, information, theory, etc.) or an event (e.g., a republic day parade). We are discussing below the major components of a system.

1. ***Inputs:*** Inputs are the basic components that enter the system for processing. Data in the form of documents, manual or computer-based files, human voice, pictures and other media are the major inputs of a system. Information and software are other forms of inputs for a computer-based information system.

2. ***Outputs:*** After processing the inputs, the system produces the desired outputs in the form of documents, reports, things, materials and ideas.

3. ***Processors:*** Processor is the operational component of a system that processes the inputs to produce the desired outputs. People and computers are the examples of processors for manual and computer-based information system, respectively.

4. ***Control and Feedback Measures:*** A system must have adequate control and feedback measures to monitor the outputs of the system and to compare with the system goals. Control is a subsystem that controls the activities of inputs, outputs and processors. Feedback measures help the system in achieving control. We will further discuss the control and feedback measures of computer-based information systems on System Control and Reliability.

5. ***System Boundary and Environment:*** All systems have boundaries that identify their components and subsystems during their interface with other systems. For instance, the components and subsystems of a computerized MIS are identified by its boundary as illustrated in Figure 1.3 showing components of a system. Anything outside the system boundary is called the system environment.

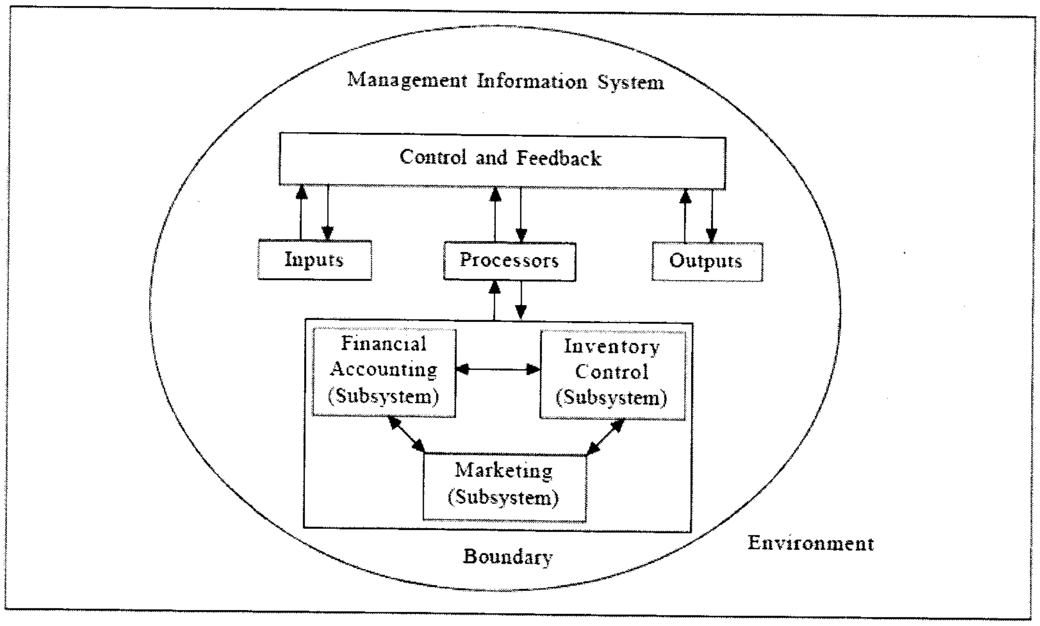

Management Information System
Control and Feedback
Inputs
Processors
Outputs
Financial Accounting (Subsystem)
Inventory Control (Subsystem)
Marketing (Subsystem)
Boundary
Environment

Figure 1.3: A Computerized Management Information System Sharing Components of a System

1.11 BUSINESS SYSTEMS

A system is a set or arrangement of interdependent things or components that are related, form a whole, and serve a common purpose. There are two types of systems:

- Natural
- Fabricated

The solar system and the human body are natural systems. They exist in nature. Fabricated systems, on the other hand, must be built by people, for example, a manufacturing system, an accounting system, and an information system.

A business organization can also be viewed as a system. The obvious advantage of applying systems' principles on a business is that then all the established and tested means and ways of a system can be applied on a business just as on any system.

The components of a typical business system are:

- Marketing
- Manufacturing
- Sales
- Research & Development
- Shipping and Logistics
- Accounting
- Personnel
- Management

All these components work together to create profit that benefits the employees and shareholders of the firm. Each of these components in itself can be viewed as systems themselves - called subsystems - of the higher-level system - the business organization. For example, the accounts system may consist of accounts payable, accounts receivable, billing, auditing, and so on.

1.11.1 Characteristics of Business

Though businesses differ greatly in their form and operations, here are the general characteristics of a business.

- Sale, transfer or exchange for the satisfaction of human needs
- Dealings in goods and services
- Recurrence of transactions
- Profit motive
- Elements of risk

Pictorially the same is illustrated below where the double arrows represent control and feedback.

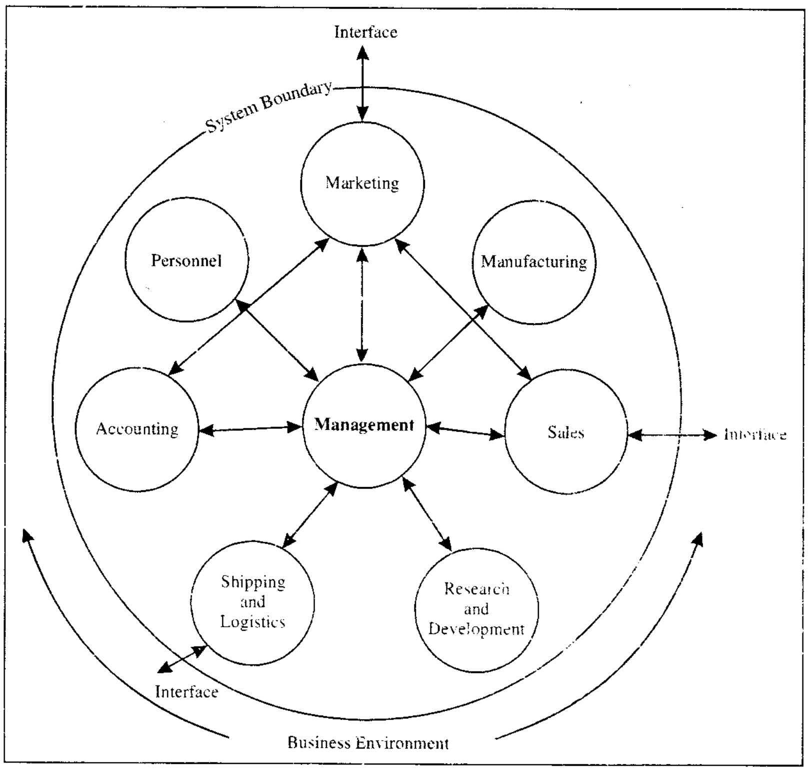


Figure 1.4

In other words "an organized production or sale of goods undertaken with the objective of earning profits through the satisfaction of human wants, is known as business."

Organizations consist of many business systems, which have similarities across different types of organizations. Their purpose is to produce goods or products that fulfill a demand in the market.

To achieve this objective, the systems interact with their environments to acquire the necessary materials, workers, and knowledge to manufacture the goods. The manufacturing systems also produce outputs, such as finished products, waste, and production technology.

To keep functioning, manufacturing systems must meet performance standards. In reality, manufacturing systems are self-regulating and self-adjusting, i.e., they indicate when:

- Personnel must be replaced
- New equipments must be purchased
- Procedures must be modified, etc.

If internal controls are not satisfactory, quality goes down dangerously or prices get unreasonably high.

Manufacturing systems are themselves subsystems within large organizations and are in turn made up of other subsystems such as:

- Materials management
- Maintenance management
- Training and development

The general features of all systems are identical. Any system can be examined with this framework in mind and specific details can be added as necessary. It is this flexibility that makes systems concept so useful to business in general and to information systems in particular.

1.12 SYSTEM MODELS

It is often convenient to analyse, design and study a system, if we concentrate on its key features rather than on its details. A model is a representation of key features of a system in various forms. It shows various elements of a system alongwith their relationships. For example, prior to designing an inventory control system, it can be represented in the form of diagrams and mathematical equations in order to visualise its structure and functions.

1.12.1 Types of Systems Models

A system model can be in the form of a diagram, mathematical equations, narrations or a physical entity as illustrated in Figure 1.5.

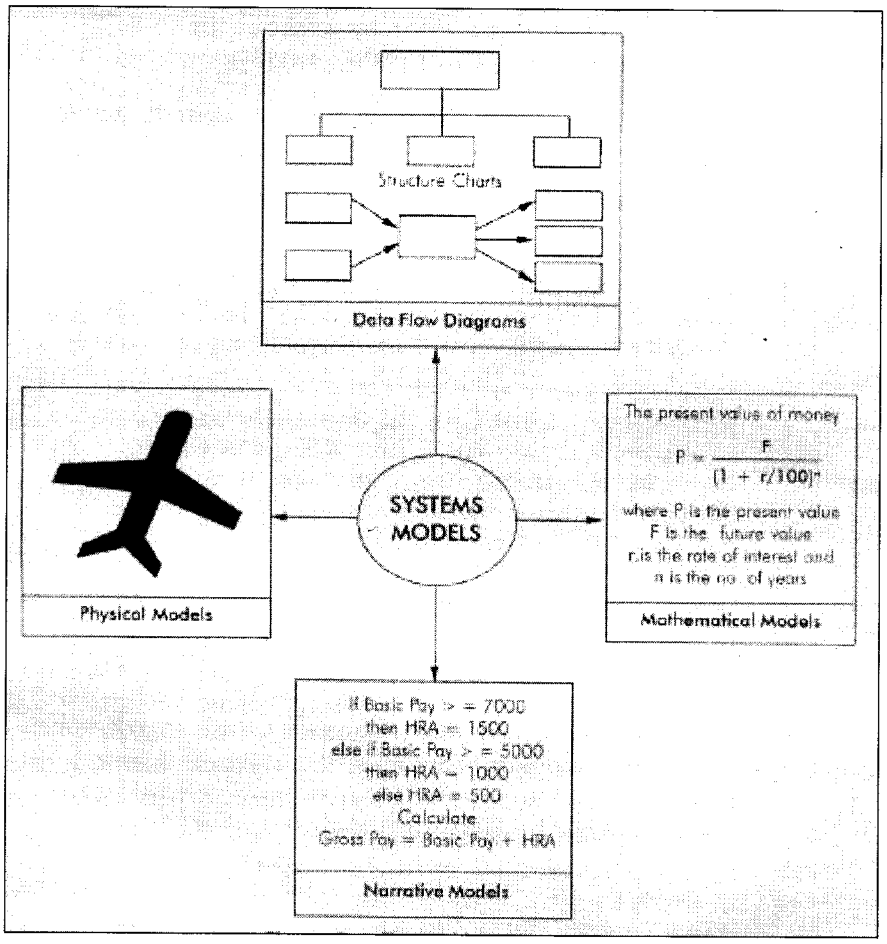


Figure 1.5: Types of Systems Models

1. ***Graphical Models:*** Graphical models represent a system in the form of diagrams by using various symbols such as boxes, lines, icons, circles etc. An information system is designed by drawing data flow diagrams (DFDs) and structure charts.

2. ***Mathematical Models:*** Mathematical models express those phenomena of the system in mathematical terms that uses mathematical computations. For example, the mathematical model for calculations of reorder level in an inventory control system is expressed in Figure 1.5.
3. ***Narrative Models:*** Narrative models describe the system in a written or spoken words in the form of process descriptions, a statement and audio/video tapes. For example, the procedures of an information system are described by Structured English (pseudocodes).
4. ***Physical Models:*** Physical models describe the system in the form of a physical entity. For example, an aeroplane can be represented in the form of a working model prior to its designing and manufacturing. A prototype (working model of hardware or software system) is another example of a physical model.

Check Your Progress

1. Fill in the blanks:
   (a) Physical Systems are also called .... systems.
   (b) The various elements of a system are interacted with others to achieve a common
   (c) Systems composed only of people are called .... systems.
   (d) Control is a .... that controls the activities of inputs, outputs and processors.
   (e) .... models describe the system in a written or spoken words.
2. State true or false:
   (a) Abstract systems are concerned with theoretical structures that may or may not exist in real life.
   (b) Almost all systems are open.
   (c) MIS system is an example of an adaptive system.
   (d) All systems have boundaries that identify their components.
   (e) A prototype is an example of graphical model.

1.13 LET US SUM UP

System analysis and design refers to the process of examining a business situation with the intent of improving it through better procedures and methods. There is a difference between system analysis and system approach. System approach applies procedural and scientific methods to observe, clarify, identify and solve a problem whereas system analysis is a management technique that helps in designing a new system or improving an existing system.

A system has a number of characteristics like organization, interaction, interdependence, Integration and central objective. It also has four basic elements like outputs and inputs, files, processor(s) and feedback.

System can be classified into various types as physical or abstract system, open or closed system, social machine or people-machine system, manual or computer-based system, stationary or non-stationary system and adaptive or non-adaptive system.

A system is composed of various components such as inputs, outputs and processors. A system must have adequate control and feedback measures to monitor the outputs and system goals. All Systems have boundaries to identify their components and subsystems during their interface with other systems.

A system model is a representation of key features of a system in various forms. Basic types of system model include graphical model, mathematical model, narrative model and physical model.

1.14 KEYWORDS

System: A Coherent set of interdependent component which exists for some purpose, has some stability and can be usefully viewed as a whole, generally portrayed in terms of an input-process-output model existing within a given environment.

Environment of a System: Anything outside a system which has an effect on the way the system operates.

Sub-system: A part of the system that carries one part of the system function.

System Study: A study of the operations of a set of connected elements and of the inter-connections between these elements.

System Analysis: Process of gathering and interpreter facts, diagnosing problems and using the information to recomment improvement to the system.

System Design: The process of planning a new system or replace or complement an existing system.

System Approach: A set of procedure for solving a particular problem, to optimize the net results of the operations of an organization.

Physical System: Systems that are made of physical entities, viz., people materials, machines, energy, etc. and intangible things viz., information, ideas, policies, procedures, etc.

Abstract System: Systems composed of only ideas, policies and theories and not of any physical entity.

Open and Closed System: An open system is that which interacts with its environment by receiving inputs from and sending outputs to outside the system. A close system is that which does not interact with its environment.

Information System: A people-machine sub-system of the business system that supports the operational, managerial and decision-making information needs of an organization.

Executive Support System: System designed to address unstructured decision making at the strategic level of an organization.

Management Information System: System designed to serve the functions of planning, controlling and decision-making at the management level of an organization.

Decision Support System: System that serves the information needs at management level of an organization.

Office Automation System: System that serves the knowledge level of an organization for supporting knowledge workers like production managers, EDP managers, etc.

Transaction Processing System: System designed to serve the operational level of an organization.

1.15 QUESTIONS FOR DISCUSSION

1. What is the basic difference between "System approach" and "System analysis"?
2. Discuss the important characteristics of a System with Suitable examples.
3. What are the basic elements of System Analysis?
4. Differentiate between the following systems (with examples):
   (a) Physical and Abstract Systems
   (b) Open and Closed Systems
   (c) Machine and People-Machine Sysems
   (d) Manual and Computer-based Systems
   (e) Adaptive and Non-Adaptive Systems
5. What are the various components of a System?
6. Define System Model.
7. Define the following terms:
   (a) System (b) Subsystem
   (c) Systems Approach (d) Systems Analysis
   (e) Systems Design
8. What is an information system? Explain the different types of information systems.
9. What is a system model? Discuss the importance of various types of models.

Check Your Progress: Model Answers

1. (a) Empirical system
   (b) Objective
   (c) Social Systems
   (d) Sub-system
   (e) Narrative
2. (a) True
   (b) True
   (c) False
   (d) True
   (e) False

1.16 SUGGESTED READINGS

Lars Skyttner, *General Systems Theory*, World Scientific.

Ludwig von Bertalanffy, *General System Theory : Foundations, Development, Applications*, New York : George Brazillier, 1969.

LESSON

2

SYSTEM DEVELOPMENT LIFE CYCLE

CONTENTS

2.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Define system development life cycle
- Describe various phases of SDLC

2.1 INTRODUCTION

The System Development Life Cycle (SDLC) is the traditional system development method used by most organizations today. The SDLC is a structured frame that consists of sequential processes by which information systems are developed. These include investigation, system analysis, system design, programming testing, implementation, operation and maintenance. These processes, in turn, consist of well-defined tasks. Some of these tasks are present in most projects, whereas others are present in only certain types of projects. That is, large projects typically require all the tasks, whereas, smaller development projects may require only a subset of the tasks.

In the past, developers used the 'Waterfall approach' to the SDLC in which tasks in one stage were completed before the work proceeded to the next stage Today, system developers go back and forth among the stages as necessary.

This lesson presents the concept of the Systems Development Life Cycle (SDLC), which provides the framework for all activities in the development process.

2.2 DEFINITION OF SDLC (SYSTEM DEVELOPMENT LIFE CYCLE)

System development for business applications is not an easy task. In developing a large integrated system such as MIS, many people are involved and many months or even years are spent. However, a small independent application such as Payroll can be developed in few weeks or months by a single or few programmers. For such small systems, system development activities may be done implicitly without proper documentation. But, for large systems, these activities must be done explicitly with proper planning and documentation. Whether a system is small or large, system development revolves around a life cycle that begins with the recognition of users' needs and understanding their problems. Such a life cycle comprising various phases is called System Development Life Cycle (SDLC).

2.3 PHASES OF SDLC

System development begins with the recognition of user needs followed by a sequence of activities which are performed step by step. The basic activities or phases that are performed for developing a system are:

1. Feasibility Analysis (Preliminary Investigations)
2. System Analysis (Requirements Analysis) and Project Planning
3. System Design
4. System Coding
5. System Testing
6. Implementation
7. Maintenance

The various phases of SDLC are illustrated in Figure 2.1.

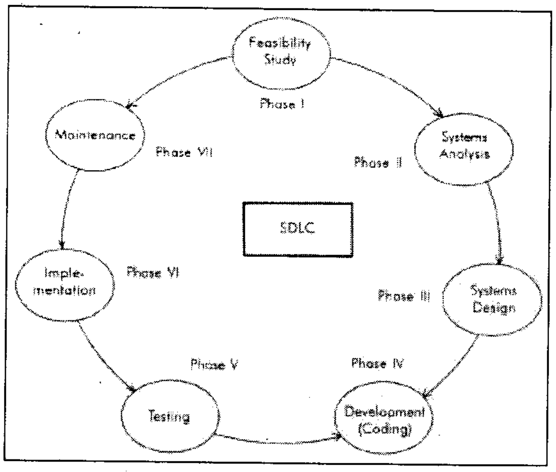


Figure 2.1: Seven Phases of System Development Life Cycle

2.3.1 Feasibility Analysis

Feasibility analysis is the first phase in the development of a new system (candidate system). This phase starts when the user faces a problem in the current system (manual or already computerised) and hence, recognizes a need for improving an information system. Some of the examples for recognizing a need are:

- The accountant of a company may feel that it is very time consuming to maintain the account books manually. He may not be able to prepare the financial statements on time due to slow manual operations.
- The management may recognize the need for getting MIS report for effective decision-making and financial planning.
- If an information system is already computerised, it may become slow and inefficient for meeting new demands of the users.
- As the information technology is evolving very rapidly, the application software developed few years back in one operating environment (say DOS or UNIX) may become outdated due to popularity of graphical operating environments such as Windows.
- Due to the Y2K problem, systems developed during the nineteenth century either may not work or may provide unexpected and incorrect information.

There may be many more reasons due to which users either want to develop new system or improve the existing system. Some of the reasons for needing system change recognized by different groups of people within and outside the organisation are summarized in Table 2.1.

Table 2.1: Reasons for Needing System Change

	Source of Change	Reason for Change
Organisation-based	Management	Unable to get MIS reports required for decision-making and planning.
	Administrative Staff information system.	Unable to handle the workload due to manual or outdated
	Sales & Marketing Staff	(a) Unable to achieve sales targets due to inefficient marketing system. (b) Unable to process orders and handle despatch on time due to inefficient invoicing/order processing system.
	Production Staff	Unable to manufacture items on schedule and with required quality due to lack of computerised production planning and control system.
	Purchase Department	Unable to control inventory of items due to inefficient inventory control system.
	Accountant	Unable to maintain account books and prepare financial statements on time.
Information-based	Customer	Slow, inefficient and unsatisfactory public dealings and consumer services provided by the organisation.
	Government	Implementation of new or revised rules and regulations imposed on the organisation.
	Other Organizations	Competing in the market for best products and services.
	Union	Settlement of disputes regarding provident funds, revision of salaries and increments, etc.

After recognizing need for system change, the user submits a formal request to the Information Systems (IS) department of the organisation or an outside software development company either for a new system or for modifying the current system. After receiving the request, the overall incharge of software development team, System Analyst, begins the preliminary investigations to determine whether the system requested is feasible to develop or not. During preliminary investigations, the analyst understands the major requirements of the system by meeting with the users. He asks a number of questions to the user regarding the current working of the system, problems of the current system and their present needs. All these activities are a part of feasibility analysis phase. At the end of feasibility analysis, a business proposal or a feasibility study report is put forth by outsider software company or in-house IS department. On acceptance of the proposal, the SDLC enters into the phase of system analysis and project planning.

2.3.2 System Analysis and Project Planning

When the System analyst decides that the requested system is feasible and the management agrees to continue the development process, SDLC enters into its next phase determination of system requirements. This phase includes studying of existing system in detail and collecting data in order to find out the requirements of the users.

This phase is also called Requirements Analysis. The major objective of requirements analysis is to identify 'what' is needed from the system. During this phase, the analyst identifies the detailed requirements of the users. At this stage, he does not emphasizes on 'how' the system will meet these objectives. System or Requirements analysis phase consists of two sub-phases - Problem Analysis and Requirements Specification. In problem analysis, the analyst understands the existing system for finding the requirements of the proposed new system (sometimes also called candidate system). In requirements specifications, the analyst specifies all the requirements on a document called Software Requirements Specifications (SRS) document. The SRS is validated by the users by reviewing their requirements specified during analysis. We will discuss the techniques and tools used for analysis in subsequent units of the book.

After completion of SRS document, the analyst makes a plan to manage the software project. System planning is the most essential part of analysis phase. During system planning, the total cost of developing the software is estimated along with the total duration of the project. A project team is organized with a detailed staff requirement for each phase. How a software project is planned? What are the major activities of systems planning? We will discuss about all these concepts later in this book.

2.3.3 System Design

After successful completion of system analysis and planning, the system is designed. In system design phase, first the system is broken down into different modules and then its each module is designed. The various activities of system design includes identification of input data and output reports, design of input forms and output forms, design of data files and developing various procedures to process the data. The major objective of system design is to produce the best model of the system as per the requirements of the users. During this phase, the analyst also studies the working environment and designs the system accordingly. A system is designed on the basis of many principles and according to a methodology. The users and managers also put certain restrictions in designing the system. All these important issues will be discussed dealing with Modularization and Module Specification. The various design reports are the outputs of system design phase.

2.3.4 System Coding

The next phase is concerned with translating the system design specification developed thus far into a working entity. Thus, if the system is mechanical, parts are procured and assembled according to the design specification.

When the design (properly documented) is accepted by the requested department, the analyst begins developing the software using a programming language. This is the phase when the programmers play their role in development of the system. They start designing data structures and writing of programs as per the documents prepared during design phase. They test their individual programs and integrate them into a single system. The development phase can be categorized into two sub-phases - Database design and Program design. Database design is the most important aspect of developing a new system. Program design is mainly concerned with writing of programs (coding), editing of programs using a text editor or word processor, debugging and finally testing them. During this phase, a team of programmers work under guidance of their project leader (systems analyst) and do all the coding.

2.3.5 Systems Testing

Testing is the most vital phase of SDLC. In this phase, the system as a whole is tested with different techniques to ensure that the system is bug free. Although, during design phase, the programmers test their programs but this sort of testing is generally unorganized without preparation of test data. During testing phase, the testing is done in systematic and organized way in order to ensure the reliability of the system and to make it error free. The major objective of testing is to find all possible errors that can occur during implementation of the system.

2.3.6 Implementation

After testing, the system is installed at the user's place and implemented. Implementation is the most crucial phase of SDLC. During implementation process, the manual or old computerised system is converted to newly developed computerised system. Many different activities are performed in conversion process on Systems Implementation and Maintenance. After conversion, the users are trained for operating and maintaining the system.

Implementation is generally considered the last phase of SDLC. However, the systems development work continues until the users of the requested department accept the candidate system.

2.3.7 Maintenance

After implementation, the systems need to be maintained in order to adapt to the changing business needs. Maintenance is sometimes not considered as a phase of SDLC, but it is an essential part of a software project that never ends. Generally more than 50 percent of total system development time is spent in maintaining the system. This is because if the system is not properly maintained, it may fail. Maintenance includes the activities that ensure to keep the system operational even if it is required to be modified later. The activities of maintenance phase can be divided into two broad categories - Corrective maintenance and Adaptive maintenance. If there are bugs (errors) in the system, they are corrected by the activities of corrective maintenance. If certain modifications are required in the system, they can be incorporated into the system by the activities of adaptive maintenance. Maintenance phase is also considered as the most expensive phase of SDLC.

Although, a system is developed very systematically with all precautions, there is always a possibility that the candidate system fails due to any major mistake made in any of the development phase. In that case, any or all of the phases are needed to be reviewed again, so that the system is completely accepted by the requested department. This is the reason, why 'life cycle' term is used in system development phases. We will discuss about each phase in detail in subsequent units.

Check Your Progress

1. What is Feasibility Analysis?
2. Define maintenance phase of SDLC.

2.4 LET US SUM UP

System development revolves around a-life cycle that begins with the recognition of users needs and understanding their problems. Such a life cycle comprising various phases is called System Development Life Cycle (SDLC). System development process is divided into several phases such as feasibility Analysis, System Analysis, System Design, System Coding, Testing, Implementation and Maintenance.

2.5 KEYWORD

System Development Life Cycle: Whether a system is small or large, system development revolves around a life cycle that begins with recognition of users' needs.

2.6 QUESTIONS FOR DISCUSSION

1. What is System Develpment Life Cycle? How does it related to System Analysis?
2. List various phases of SDLC.
3. Give any five reasons due to which users want to develop new system or improve existing system.
4. What do you mean by project planning?
5. Define System Design.
6. Define System Coding. Name its sub faces.
7. Why is System testing required?
8. List the various categories of maintenance face.

Check Your Progress: Model Answers

1. Feasibility analysis is the first phase in the development of a new system (candidate system). This phase starts when the user faces a problem in the current system (manual or already computerised) and hence, recognizes a need for improving an information system.
2. After implementation, the systems need to be maintained in order to adapt to the changing business needs. Maintenance is sometimes not considered as a phase of SDLC, but it is an essential part of a software project that never ends.

2.7 SUGGESTED READINGS

Avdesh Gupta, Anurag Malik, *Management Information Systems*, Firewall Media.

Michael Bronzite, *System Development*, Springer.

Preeti Gupta, *System Analysis and Design*, Firewall Media.

LESSON

3

CASE TOOLS AND SYSTEM ANALYSIS

CONTENTS

3.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss definition of CASE tools and their building blocks
- Discuss taxonomy of CASE tools
- Understand Integrated CASE environment
- Discuss the CASE repository
- Define a system analyst
- Describe the role of system analyst in SDLC
- Describe the qualities that a system analyst should attain

3.1 INTRODUCTION

Computer Aided Software Engineering (CASE) tools assist the software engineering managers and practitioners in every activity that is a part of the software process. They automate project management activities, manage all work products produced throughout the process and assist engineers in their analysis, design, coding and testing. CASE tools can be integrated within a sophisticated environment.

The system analyst is overall responsible for the development of a software. He is the crucial interface between users, programmers and MIS managers. He conducts a system's study, identifies activities and objectives and determines a procedure to achieve the objective. He has a very important role in the development of a system. The concerned person should also have some special qualities which we are going to discuss in this lesson.

3.2 CASE: DEFINITION

A workshop of any artisan e.g. carpenter, mechanic, software engineer, etc. must have three characteristics: (1) a collection of tools that will help in every step of building a product, (2) an organized layout that enables tools to be found quickly and used efficiently and (3) a skilled artisan who understands how to use these tools in an effective manner.

The workshop for software engineering has been called an integrated project support environment and the tools that fill the workshop are collectively called computer-aided software engineering.

CASE provides the software engineer with the ability to automate manual activities and to improve project insight. CASE tools ensure that quality is designed before the product is built.

3.3 BUILDING BLOCKS FOR CASE

Computer aided software engineering can be a single tool or a complex complete environment that comprises of tools, database, people, hardware, network, operating systems, standards and numerous other components. The building blocks of CASE are illustrated in Figure 3.1. Each building block acts as a foundation for the next one. Actually, an effective CASE foundation has nothing to do with the software engineering tools themselves. These successful environments for software engineering are built on an architecture that includes both hardware and systems software. Also, the environment architecture must also include the human network patterns that are applied during the software engineering process.

The environment architecture comprises of hardware platform and system support and acts as the ground work of the CASE. But this environment in turn is composed of building blocks. A set of portability services provides a bridge between CASE tools and their integration framework and the environment architecture. The integration framework is a collection of specialized programs that enables individual CASE tools to communicate with one another to create a project database, and to exhibit the same look and feel to the end-users. Portability services allow the CASE tools and their integration framework to migrate across different hardware platforms and operating systems without significant adaptive measures.

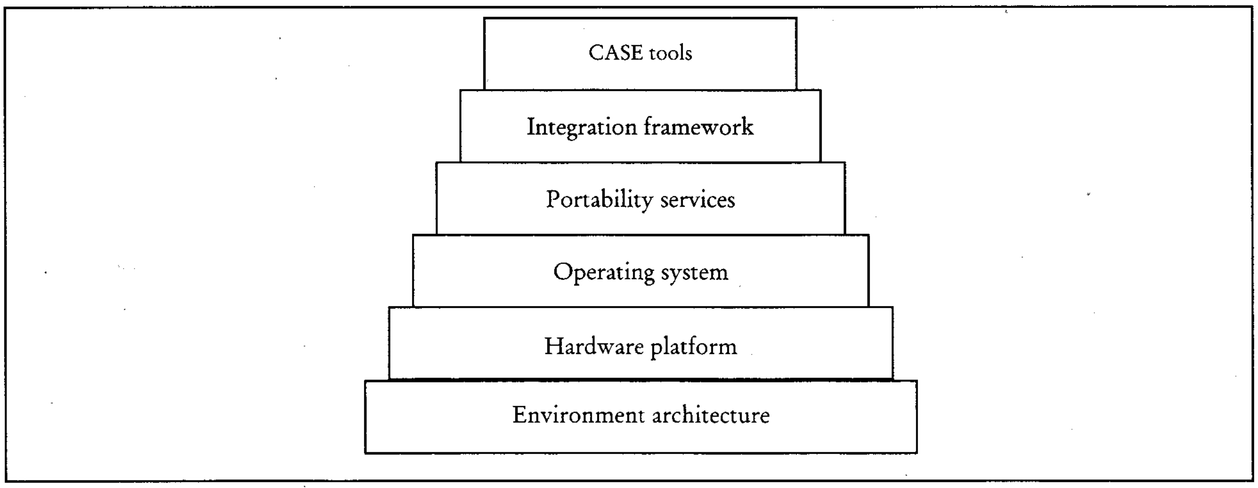


Figure 3.1: CASE Building Blocks

The building blocks in Figure 3.2 represent a comprehensive foundation for the integration of CASE tools. However, most CASE tools that are being used today are not made of these building blocks. Some CASE tools are used in a particular software engineering activity and do not communicate directly with other CASE tools, are not bound to a database and are not parts of the integrated CASE environment.

The relative levels of integration are shown in Figure 3.2. At the low end of the integration spectrum is the individual tool. When individual tools exchange data the integration level is improved. Such tools produce output in such a format that is acceptable to other tools. At times the builders of these tools come up with bridges between these tools. The bridge approach will produce the end products which will be difficult to create using either tool separately. Single-source integration occurs when a single CASE tools vendor integrates multiple different tools and sells them as a package. Although this is an effective approach but it sometimes prohibits the easy addition of tools from other vendors in their environment.

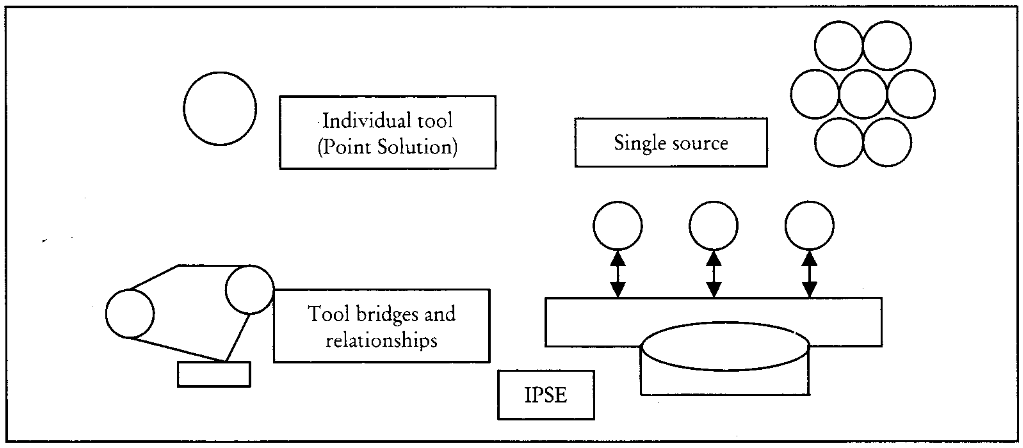


Figure 3.2: Integration Options

At the high end of the spectrum is the Integrated Project Support Environment (IPSE). CASE tools vendors use IPSE standards to build tools that will be compatible with the IPSE and therefore compatible with one another.

3.4 A TAXONOMY OF CASE TOOLS

A number of risks can creep in if we attempt to categorize CASE tools. Confusion can be created by placing a tool within one category when others believe that it belongs to another category. Although with risks, but it is essential to create a taxonomy of CASE tools to understand the breadth of CASE and to better appreciate where such tools can be applied in the software engineering process.

CASE tools can be classified on the basis of function, role, use in various steps, environment architecture or origin or cost. Here we will discuss the taxonomy on the basis of function.

1. ***Business process engineering tools:*** These tools provide a "meta-model" by modeling the strategic information requirements of an organization from which the specific information systems are obtained. Rather than focusing on individual component requirements the business information is modeled as it moves between various organizational entities within a company. The main aim of the tools in this category is to represent business data objects, their relationships, and how these data objects flow between different business areas within a company.

2. ***Process modeling and management tools:*** It is must for an organization to understand the business process in order to improve on it. Process modeling tools or process technology tools, are used to represent the key elements of a process so that it can be better understood. These tools also provide link to process descriptions to those who are involved in the process to understand the work tasks that are required to perform it. They provide links to other tools that provide support to defined process activities.

3. ***Project planning tools:*** Tools belonging to this category focus on two primary areas: software project effort and cost estimation and project scheduling. Estimation tools calculate the estimated effort, duration and recommended number of people for a project. Project scheduling tools enable the manger to define all project tasks, create a task network, represent task inter dependencies and model the amount of parallelism possible for the project.

4. ***Risk analysis tools:*** Identifying potential risks and developing a plan to lessen, monitor and manage them is of utmost importance in large projects. Risks analysis tools enable a project manager to build a risk table by providing detailed guidance in the identification and analysis of risks.

5. ***Project management tools:*** The project schedule and project plan must be tracked and monitored on a continuous basis. Also, a manager should use tools to collect metrics that will ultimately provide an indication of software product quality. Tools in the category are often extensions to project planning tools.

6. ***Requirement tracing tools:*** At times it happens that when a large system is developed and delivered it does not meet the customer requirements. The aim of the requirements tracing tools is to provide a systematic approach to the isolation of requirements, beginning with the customer request for proposal or specification. The typical requirements tracing tool combines human interactive text evaluation with a database management system that stores and categorizes each system requirement that is parsed from the original RFP.

7. ***Metric and management tools:*** Software metrics improve a manager's ability to control and coordinate the software engineering process and a practitioner's ability to improve the quality of the software being produced. Management-oriented tools capture project specific metrics like LOC/person-month, defects per function point, etc that provide an overall indication of productivity or quality. Technically oriented tools determine technical metrics that provide greater insight into the quality of design or code.

8. ***Documentation tools:*** These types of tools support each aspect of software engineering. Documentation tools provide an important opportunity to improve productivity of the software practitioners as 20 to 30 percent of the software development time is spent on documentation.

9. ***System software tools:*** CASE is a workstation technology, thus, it must include high quality network systems software, object management services, distributed component support, e-mail, bulletin boards and other communication capabilities.

10. ***Quality assurance tools:*** These tools are nothing but metric tools that audit source code to determine compliance with language standards. Other tools draw technical metrics in order to project the quality of the software being built.

11. ***Database management tools:*** Database management software serves as a foundation of the CASE repository called the project database. CASE tools for database management have evolved from relational database management systems to object oriented database management systems.

12. ***Software configuration management tools:*** Software configuration management lies at the core of every CASE environment. These tools can help in all the SCM related tasks: identification, version control, change control, auditing and status accounting.

13. ***Analysis and design tools:*** These tools enable a software engineer to create models of the system to be built. The model contains a representation data, function and behavior and characteristics of data, architectural, component-level and interface design. These tools help the software engineer to have a better insight into the analysis representation, through consistency and validity checking, to help eliminate errors before they propagate into the design or build.

14. ***Programming tools:*** This category comprises of compilers, editors, debuggers that are available to support most conventional programming tools. In addition to these, it includes the object-oriented programming tools, fourth generation languages, graphical programming environments, etc.

15. ***Prototyping tools:*** A variety of prototyping tools can be used. Screen painters enable a software engineer to define screen layout rapidly for interactive applications. More sophisticated CASE prototyping tools enable the creation of a data design, coupled with both screen and report layouts. Many analysis and design tools have an extension in the form of a prototyping tool. PRO/SIM tools generate skeleton Ada and C source code for developing real-time applications.

16. ***Integration and testing tools:*** The SQE defines the following testing tools categories in its directory of testing tools:

    (a) *Data acquisition:* tools that acquire data to be used during testing.

    (b) *Static measurement:* tools that analyze the source code without executing test cases.

    (c) *Dynamic measurement:* tools that analyze source code during execution.

    (d) *Simulation:* tools that simulate function of hardware or other externals.

(e) *Test management:* tools that assist in the planning, development and control of testing.

(f) *Cross-functional tools:* tools that cross the bounds of the preceding categories.

Many testing tools have features spanning across two or more categories.

3.5 INTEGRATED CASE ENVIRONMENT

Although a lot of benefit can be derived from individual CASE tools that cater to each software engineering activity separately, the real power of CASE can be achieved only through integration. The benefits of integrated CASE (I-CASE) include:

- Smooth transfer of information from one tool to another and one software engineering step to another.
- A reduction in the effort required to perform umbrella activities such as SCM, SQA and documentation production.
- An increase in project control that is achieved through better planning, monitoring and communication.
- Improved coordination among staff members who are working on a large software project.

However, I-CASE also poses certain problems in terms of integration demanding consistent representation of software engineering information, interfaces and communication. Although I-CASE environments are evolving slower than expected but they are becoming more powerful with years passing by.

To define integration in the context of software engineering process, it is necessary to establish a set of requirements for I-CASE. An integrated CASE environment should possess the following as per Pressman:

1. Provide a mechanism for sharing software engineering information among all tools contained in the environment.
2. Enable a change to one item of information to be tracked to other related information items.
3. Provide version control and overall configuration management for all software engineering information.
4. Allow direct, non-sequential access to any tool contained in the environment.
5. Establish automated support for the software process model that has been chosen, integrating CASE tools and Software Configuration Items (SCI) into a standard work breakdown structure.
6. Enable the users of each tool to experience a consistent look and feel at the human/computer interface.
7. Support communication among software engineers.
8. Collect both management and technical metrics that can be used to improve the process and the product.

In order to achieve this all the components must be put together in a harmonious way by adhering to the integration architecture explained below.

3.5.1 The Integration Architecture

A software engineering team creates a pool of software engineering information out of CASE tools, corresponding methods and a process framework. The integration framework allows transfer of information in and out of this pool. In order to achieve this, the following should exist: a database must be created; an object management system must be built, a tools control mechanism must be built, a user interface must be provided to ensure a consistent pathway between actions made by users and the tools contained in the environment. A simple model depicting the framework is shown in Figure 3.3.

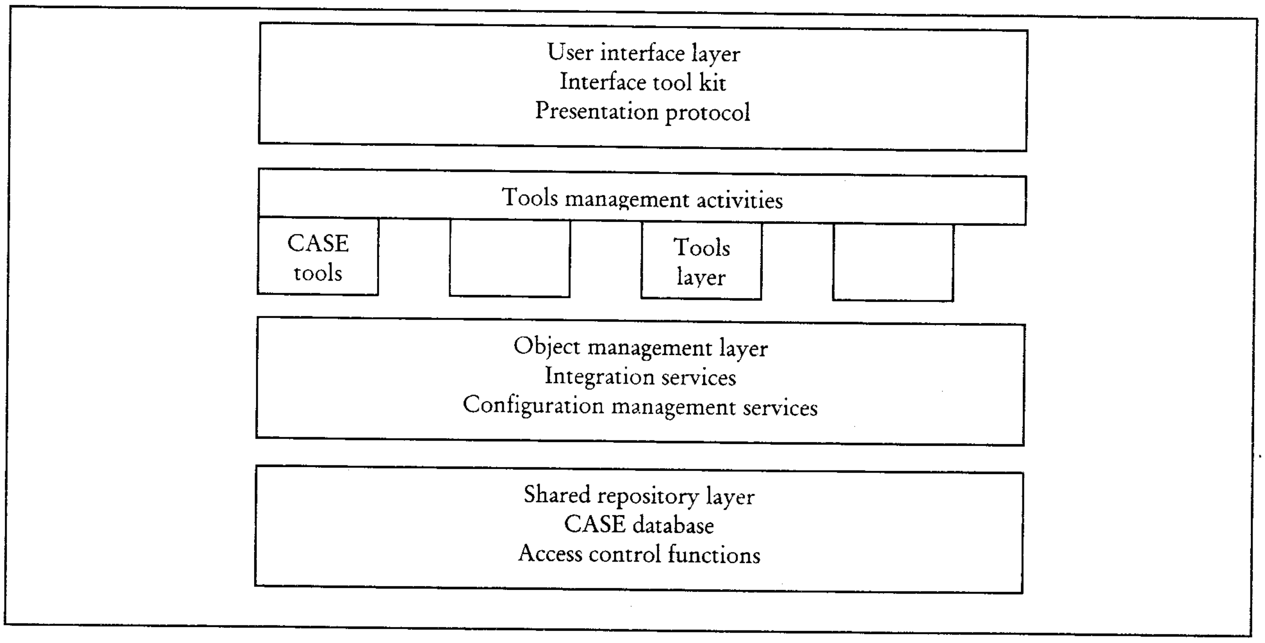


Figure 3.3: Architectural Model for the Integration Framework

The *user interface layer* includes a standardized interface tool kit with a common presentation protocol. The interface tool kit contains software for human/computer interface management and a library of display objects. The presentation protocol is the set of guidelines that gives all CASE tools the same look and feel. Screen layout conventions, menu names and organizations, object names, the use of the keyboard and mouse are all defined as part of the presentation protocol.

The *tool layer* comprises of tools management services with the CASE tools themselves. Tools Management Services (TMS) control the behavior of tools within the environment. If multitasking is used, TMS performs multitask synchronization and communication, coordinates the flow of information from the repository and object management system into the tools, etc.

The Object Management Layer (OML) performs the configuration management functions and the tools integration mechanism. Every CASE tool is plugged into the OML. Thus, the OML provides the integration services in conjunction with the CASE tools.

The shared repository layer is the CASE database and the access control functions that enable the object management layer to interact with the database. Data integration is achieved by the object management and shared repository layers.

3.6 THE CASE REPOSITORY

The CASE repository refers to the storage of mechanisms, tools and data structures used for various software engineering activities. The repository of any I-CASE environment is the set of mechanisms that achieve data/tool and data/data integration. In addition to the database management functions, the repository performs the following:

1. ***Data integrity*** ensures that all the entries in the repository are validated, there exists consistency among related objects and automatically performs cascading modifications when a change to one object demands changes in other related objects.
2. ***Information sharing*** provides a mechanism for sharing information among multiple developers and between multiple tools; manages and controls multiple user access to data and locks or unlocks objects so that changes are not unintentionally overlaid on one another.
3. ***Data/tool integration*** establishes a control model that can be accessed by all tools in the I-CASE environment, controls access to the data, and performs appropriate configuration management functions.
4. ***Data/data integration*** is the database management system that relates data objects so that other functions can be achieved.
5. ***Methodology enforcement*** defines an entity-relationship model stored in the repository that implies a specific paradigm for software engineering.
6. ***Document standardization*** is the definition of objects in the database that leads directly to a standard approach for the creation of software engineering documents.

To achieve these functions, the repository is defined in terms of a meta-model The meta-model determines how information is stored in the repository, how data can be accessed by tools and viewed by software engineers, how nicely data security and integrity can be maintained and how easily the existing model can be extended to accommodate new needs.

3.6.1 Features and Contents

The types of things to be stored in the repository include:

- The problem to be solved.
- Information about the problem domain.

The system solution as it emerges.

Rules and instructions pertaining to the software process being followed:

- The project plan, resources and history.
- Information about the organizational context.

3.7 WHO IS A SYSTEMS ANALYST?

A Systems analyst is a person who is overall responsible for development of a software. He is the computer professional charged with analysing, designing and implementing computer-based information systems. He is the crucial interface among users, programmers and MIS managers. A Systems analyst can be defined as follows:

A Systems analyst is a computer specialist who translates business problems and requirements into information systems and acts as liaison between IS (Information Systems) department and rest of the organisation.

The analyst conducts a systems study, identifies activities and objectives and determines a procedure to achieve the objectives. He is the key member of both MIS organisation and the software project team. He is a person with unique skills, experience, personality and common sense. His role has been emerging with advances in technology.

3.8 ROLES OF A SYSTEMS ANALYST

The Systems analyst performs the following roles during various phases of SDLC. He works as a:

(a) ***Problem Investigator:*** The analyst studies the problems and needs of an organisation during feasibility and requirements analysis phases of SDLC. He visits the various departments of the organisation and interviews the users. He analyses the problems of the current system and collects their new requirements. The analyst initially works as an investigator by extracting the real problems of the users.

(b) ***Problem Solver:*** The analyst solves the problems of the current system faced by the users. He determines how people, method and technology can improve the current system. After feasibility analysis, he presents the system proposal to the management.

(c) ***Systems Designer:*** The analyst creates a detailed physical (current) and logical (proposed) design of the system. The practical role of a systems analyst as a designer is discussed in subsequent units of the book.

(d) ***Motivator:*** The analyst motivates users to participate in development and implementation of the proposed system. This helps to understands user's feelings about the proposed system. The analyst interprets the thoughts of users and hence, draws conclusions. He appeals management and users for getting the support in development and implementation of the proposed system.

(e) ***Project Manager:*** The analyst monitors the development and implementation of software in relation to quality, cost and time. He works with the project leader for managing the project properly. For development of small systems, the Systems analyst is generally the project leader.

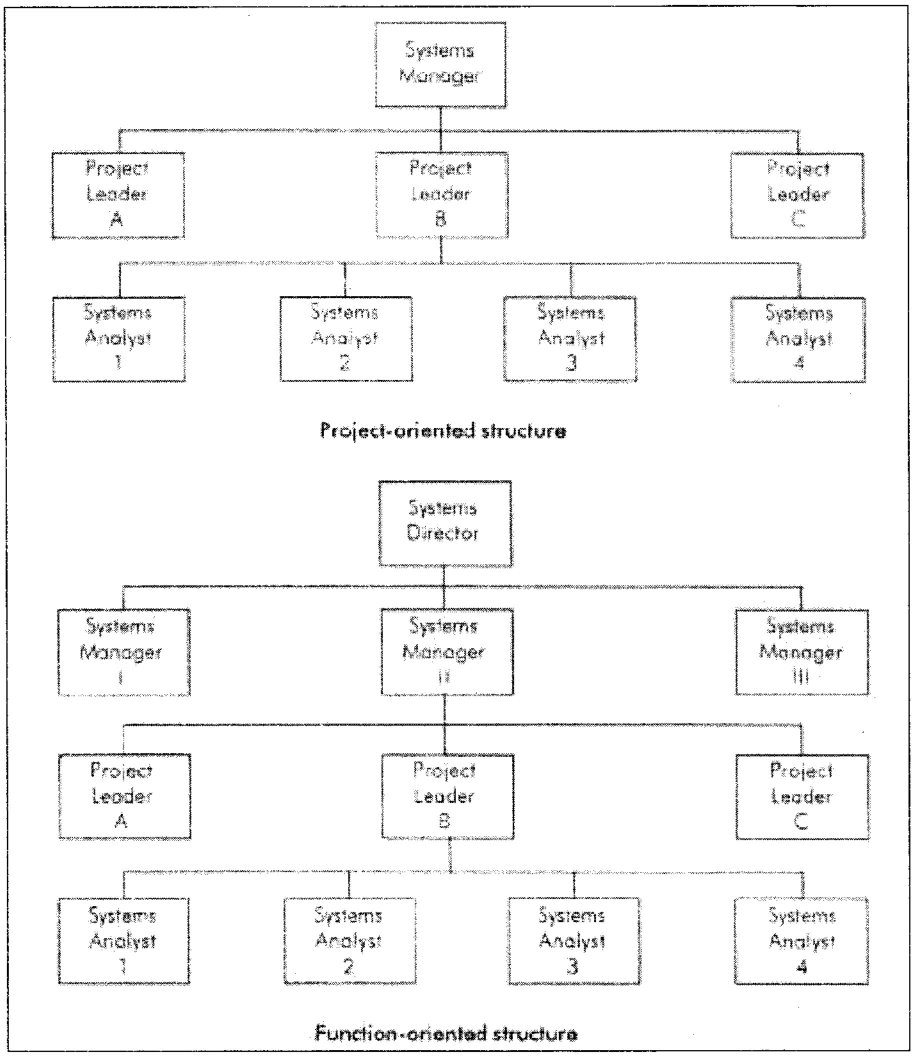


Figure 3.4: Two Types of Project Team Structures

3.9 QUALITIES OF SYSTEMS ANALYST

Success in systems analysis requires interpersonal and technical skills of the analyst. The systems analyst is expected to possess the following qualities:

(a) ***Qualified:*** The analyst must be highly qualified in software technology. Besides software, he should have a good knowledge of hardware and latest communication and networking technology. He must have a thorough awareness about the working (manual and computerised) of financial accounting, sales and marketing, invoicing, inventory control, production and other information systems of different organization.

(b) ***Analytical Thinker:*** The analyst must be capable to extract real problems of the users by analysing the existing system. He is expected to provide the best solutions to the problems. He should be able to provide more than one solution to a single problem so that the users can select the best one. The systems analyst must be capable of tackling any problem of the user. He must be a problem solver and not a problem creator.

(c) ***Good Communicator:*** The analyst must have a good communication and presentation skills. He must have an excellent command on the language which the user can understand. There should not be any communication gap between the systems analyst and users.

(d) ***Experienced:*** The analyst should be experienced in both information and management technologies. He should be associated with all types of business concerns (viz., Manufacturing, Trading, Financial, etc.). The present day systems analysts are expected to possess a good experience in development of software using 4GLS (such as Oracle, Sybase, etc.) and object-oriented languages (such as C++).

(e) ***Creator:*** The analyst should possess excellent creativity skills that help to convert ideas of the users into concrete plans. He/she should be capable of creating plans and designing systems by drawing diagrams, charts and other illustrations.

(f) ***Trainer:*** The analyst should be a good teacher for educating and training users in computer-based information systems.

Check Your Progress

1. Fill in the blanks:
   (a) The .... performs the configuration management functions and the tools integration mechanism.
   (b) .... occurs when a single CASE tools vendor integrates multiple different tools and sells them as a package.
   (c) System analyst acts as liaison between .... department and rest of the organisation.
   (d) In a project-oriented structure .... is on the top hierarchy.
   (e) In a .... structure, systems director is on the top hierarchy.
   (f) .... presents the system proposal to the management.
   (g) The analyst should be experienced in both .... and .... technologies.
2. State true or false:
   (a) Systems analyst is the key member of both MIS organisation and the software project team.
   (b) The analyst cannot be the project leader.
   (c) The analyst need not to be qualified in computer hardware technology.
   (d) The analyst must possess excellent creativity skills.
   (e) The analyst need not be a good trainer.
   (f) With the help of I-CASE an increase in project control that is achieved through better planning, monitoring and communication.

3.10 LET US SUM UP

Computer-sided software engineering tools include all the activities that are a part of the software process. CASE combines a set of building blocks that begin at the hardware level and operating system level and end with individual tools.

We learnt about the taxonomy of CASE tools. These categories comprise of both technical and management activities that span most software areas. Each category of tool is considered a point-solution.

The I-CASE environment combines integration mechanisms for data, tools and human/interface interaction. Data integration can be achieved through direct exchange of information, through common file structures, by data sharing or inter-operability or though the use of a full I-CASE repository.

The CASE repository has been referred to as a software bus. Information moves through it, passing from tool to tool as software engineering process.

A system analyst is overall responsible for development of a software according to the requirement of the management. He is a person who investigates the business problem, solves it, determines the measures to be taken to improve the current system, then creates a detailed physical and logical design of the system. He further motivates the users to participate in the development and implementation of the proposed system. At last he monitors the development and implementation of software in relation to quality, cost and time.

A system analyst must be highly qualified, must be capable to analyze the existing system to extract real problems of the users, must have good communication and presentation skill, should be experienced in both information and management technologies, should possess excellent creativity skills and he should be a good trainer for educating and training users in computer-based information system.

3.11 KEYWORDS

OML: Object Management Layer

TMS: Tools Management Services

SCI: Software Configuration Items

SCM: Software Configuration Management

SQA: Software Quality Assurance

PRO: Prototyping

SIM: Simulation

LOC: Lines of Codes

IPSE: Integrated Project Support Environment

CASE: Computer Aided Software Engineering

System Analyst: A computer specialist who translates business problems and requirements into information system.

System Designer: A computer specialist who creates a detailed physical and logical design of the system.

Project Manager: A computer specialist who monitors the development ad implementation of software in relation to quality, cost and time.

3.12 QUESTIONS FOR DISCUSSION

1. Explain in detail the CASE repository with its advantages and disadvantages.
2. Explain the term CASE with its building blocks.
3. Describe the integrated architecture of the CASE tools.
4. List three important attributes of a system analyst. .
5. Which is in your opinion the most difficult job of a systems analyst?
6. Who is Systems analyst? Discuss the position of an analyst in a MIS organization.
7. Describe the roles of Systems analyst during various phases of SDLC.
8. Success in system analysis requires interpersonal and technical skills of the analyst. Discuss why.
9. What are the qualities expected in a system analyst?
10. What are the expected qualifications of a system analyst?
11. Who is a good communicator?
12. Why is a system analyst also required to be a teacher?

Check Your Progress: Model Answers

1. (a) Object management layer
   (b) Single-source integration
   (c) Information System
   (d) System Manager
   (e) Function-oriented
   (f) System analyst
   (g) information, management
2. (a) True
   (b) False
   (c) False
   (d) True
   (e) False
   (b) True

3.13 SUGGESTED READINGS

R.S. Pressman, *Software Engineering-A Practitioner's Approach*, 5th Edition, Tata McGraw Hill Higher education.

Forte G., *In Search of Integrated Environment*, CASE Outlook, March-April 1989.

Testing Tools Reference Guide, *Software Quality Engineering*, 1995.

CASE: *The Potential and the Pitfalls*, QED Information Sciences, 1989.

Nicholas K.M., *Performance Tools*, IEEE Software, May 1990.

UNIT II

LESSON

4

REQUIREMENT ANALYSIS AND METHODOLOGIES

CONTENTS

4.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss fact finding techniques such as sampling, interviews, questionnaires, and observing the office environment, etc.
- Discuss structured system analysis techniques such as DFD, Entity Relationship Diagram (ERD), data dictionary, decision trees, decision tables, etc.

4.1 INTRODUCTION

The very first step in a system development project is to understand the requirements, the framework of the organization's objectives and the environment in which that system is going to be installed. Consideration is given to the user's resources as well as to finances.

Failure to specify system requirement before the final selection almost always results in a faulty execution of the development project. The specifications should delineate the user's requirements. They must reflect the actual applications to be handled by the system and include system objectives, flow charts, input-output requirements, file structure and cost. The specifications must also describe each aspect of the system clearly, consistently and completely.

System analysis is the first phase of system development. The focus here is to characterize the system that is supposed to evolve through the subsequent development phases. The problem space is identified and modeled using various tools and the same is captured in a document.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside the system. A key question is: What must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining, whether or not a candidate system should consider other related systems. During analysis, data are collected on the available files, decision points and transactions handled by the present system. Tools are used and logical models of the system developed.

Once analysis is completed, the analyst has a firm understanding of what has to be done. The next step is to decide, how the problem might be solved. Thus, in system design, we move from logical to the physical aspect of the life cycle.

In structured analysis and design methodology, analysis phase is subdivided into a number of sub-phases accomplishing sub-tasks of the analysis. The various tasks in analysis phase are discussed below.

4.2 PROBLEM DEFINITION

The only purpose of a system is to solve a problem. Before starting on a system development, therefore, it is extremely important that the problem itself is understood clearly and unambiguously.

Problem definition is a clear, concise and unambiguous statement of the problem. It is often not easy to arrive at a problem definition for the following reasons:

1. Users are (mostly) non-technical staff members, whereas developers are (mostly) technical people. This causes a communication gap between what is said and what is understood.
2. Immediate validation of requirements is not possible at this very stage; therefore, the developers' view of the system deviates significantly from that of the users'.
3. Development efforts are often carried out under some time and resource constraints, limiting the duration and extent of interaction between the developer and the user.

Due to the above cited reasons a system analyst's job becomes very challenging. He acts as a bridge between the two parties. In this very first stage of the very first phase of development, a problem definition is evolved and agreed upon by the two concerned parties.

4.3 INFORMATION REQUIREMENT

It is a document that serves as a foundation for hardware, software and database engineering. It describes the functions of a system and the constraints that will govern its development. The specifications bound each allocated system analyst with an indication of the role of software within the context of the computer-based system as a whole and the various sub-systems described in the data flow diagrams. The system specification also describes the information that is to input and to output from the system.

4.3.1 Information Gathering Tools

Fact-finding means, learning as much as possible about the present system. Fact-finding is the formal process of using research, interviews, questionnaires, sampling and other techniques to collect information about systems, requirements and preferences. It is also called information gathering or data collections. Tools, such as data and process models, document facts, and conclusions are drawn from facts. If you can't collect the facts, you can't use the tools. Fact-finding skills must be learned and practiced.

There are many occasions for fact-finding during the system development life cycle. However, fact-finding is most crucial to the system planning and system analysis phases. It is during these phases that the analysis learns about the vocabulary, problems, opportunities constraints, requirements, and priorities of a business and a system. Fact-finding is also used during the system design and support phases, but to a lesser extent. During system design, fact-finding becomes technical, as the analyst attempts to learn more about the technology selected for the new system. During the system support phase, fact-finding is important in determining that a system has decayed to a point, where the system needs to be re-developed.

4.4 FACT-FINDING

For fact-finding, the system analyst does the following:

1. Sampling of written (existing) documents
2. On-site observations
3. Interview
4. Questionnaires
5. Research and site visits
6. Rapid Applications Development (RAD)
7. Joint Application Development (JAD)
8. Observes the current system
9. Gather forms and documents currently in use
10. Determines the flow of data through the system, and
11. Clearly defines the system requirements.

4.4.1 Fact-finding Techniques

No two projects are ever the same. This means that, the analyst must decide on the information gathering tool and how it must be used. Although, there are no standard rules for specifying their use, an important rule is that information must be acquired accurately, methodically, under the right conditions and with minimum interruption to user personnel. For example, if the analyst needs only information available in existing manuals, then interviewing is unnecessary. Interviewing is necessary where the manual is not up-to-date. If additional information is needed, onsite observation or questionnaire may be considered. Therefore, we need to be familiar with various information gathering tools. Each tool has a special function, depending on the information needs.

Reviews of Literature, Procedures and Forms

Very few system problems are unique. The increasing number of software packages suggests that, problem solutions are becoming standardized. Therefore, as a first step, a search of the literature through professional reference and procedures manuals, textbooks, company studies, government publications, or consultant studies may prove invaluable. The primary drawback of the search is time. Often it is difficult to get certain reports, publications may be expensive and the information might be outdated due to a time lag in publication.

Procedure manuals and forms are useful sources for the analyst. They describe the format and functions of the present system. Included in most manuals are system requirements that help determine, how well various objectives are met. Up-to-date manuals save hours of information gathering time. Unfortunately, manuals are usually outdated or do not exist.

Included in the study of procedures and forms is the study of existing forms. Printed forms are widely used for capturing and providing information. The objective is to understand how forms are to be used. The following questions may be useful:

- Who uses the forms? How important are they to the user?
- Do the forms include all the necessary information? What item should be added or deleted?
- How do departments receive the existing forms? Why?
- How readable and easy to follow is the form?
- How does the information in the form help other users make better decisions? What other uses does the form offer to the user?

Onsite Observation

Another information gathering tool used in system studies is on-site observation. It is the process of recognizing and noting people, objects and occurrences to obtain information. The analyst's role is that of an information seeker who is expected to be detached (therefore, unbiased) for the system being observed. This role permits participation with the user staff openly and freely.

The major objective of any onsite observation is to get as close as possible to the real system being studied. For this reason it is important that the analyst be knowledgeable about the general make-up and activities of the system. The following questions can serve as a guide for onsite observations:

- What kind of system is it? What does it do?
- Who runs the system? Who are the important people in it?
- What is the history of the system? How did it get to its present stage of development?
- Apart from its formal function, what kind of system is it in comparison with other systems in the organization? Is it a primary or a secondary contributor to the organization? Is it fast paced or is it a leisurely system that responds slowly to external crisis?

As an observer, the analyst follows a set of rules. While making observations, he/she is more likely to listen than talk and to listen with a sympathetic and genuine interest when information is conveyed. The emphasis is not on giving advice or passing moral judgment on what is observed. Furthermore, care is taken not to argue with persons being observed or to show hostility towards one person and undue friendliness towards another.

When human observers are used, four alternative observation methods are considered:

1. ***Natural or Contrived:*** A natural observation occurs in a setting, such as the employees, place of work; a contrived observation is set up by the observer in a place like a laboratory.
2. ***Obtrusive or Unobtrusive:*** An obtrusive observation takes place, when the respondent knows he/she is being observed; an unobtrusive observation takes place in a contrived way, such as behind a one-way mirror.
3. ***Direct or Indirect:*** A direct observation takes place, when the analyst actually observes the subject or the system at work. In an indirect observation, the analyst uses mechanical devices such as cameras and videotapes to capture information.
4. ***Structured or Unstructured:*** In a structured observation, the observer looks for and records a specific action, such as the number of soups a shopper can pick up before choosing one. Unstructured methods place the observer in a situation to observe whatever might be pertinent at that time.

Any of these methods may be used in information gathering. Natural, direct, obtrusive and unstructured observations are frequently used to get an overview of the operation. The degree of structure increases, when the observation has a specific purpose. The obtrusiveness may decrease, when one wants to observe the tasks that make up a given job. Indirect observation could be used in a similar manner. Finally, contrived situations are used to test or debug a candidate system.

Electronic observation and monitoring methods are widely being used in information gathering tools because of their speed, efficiency and low cost. For example, some truck fleets use an electronic recorder system that records, analyses and reports information online about the hours and minutes a vehicle was driven faster than 60 miles/hour. Onsite observations are not without problems:

- Intruding into user's area often results in adverse reactions by the staff, therefore, adequate preparation and training are important.
- Attitudes and motivations of subjects cannot be readily observed - only the actions that result from them.
- Observations are subject to error due to the observer's misinterpretation and subjective selection of what to observe, as well as the subjects' altered work pattern during observation.
- Unproductive long hours are often spent in an attempt to observe specific one-time activities or events.

In deciding to use an onsite observation, several questions are considered.

- What behavior can be observed that cannot be described in other ways?
- What data can be obtained more easily or more reliably by observation than by other means?

- What assurance can be given that the observation process is not seriously affecting the system or the behaviour being observed?
- What interpretation needs to be made about observation data to avoid being misled by the obvious?
- How much skills are required and available for actual observation?

Onsite observation to be done properly in a complex situation can be very time consuming. Proper sampling procedures must be used to ascertain the stability. Inference drawn from small samples of behaviour can be inaccurate.

Interviews

The interview is a face-to-face, interpersonal role situation in which a person called the interviewer asks a person being interviewed, questions designed to get information about a problem area. The interview is the oldest and most often used device for gathering information in systems work. It has qualities that behavioral and onsite observations do not possess. It can be used for two main purposes:

- As an exploratory device to identify relations or verify information, and
- To capture information as it exists.

Validity is no small problem. Special pains are taken to eliminate interview bias. We assume that, information is more valid, the more freely it is given. Such an assumption stresses the voluntary character of the interview as a relationship freely and willingly entered into by the respondent. If the interview is considered a requirement, the interviewer might gain the respondents' time and attention but cannot be certain of the information gathered during the interview.

In an interview, since the analyst (interviewer) and the person interviewed meet face to face, there is an opportunity for flexibility in eliciting information. The analyst is also in a position to observe the subject. In contrast, the information obtained through a questionnaire is limited to the subjects' written responses to predefined questions.

There are four primary advantages of an interview:

- Its flexibility makes interview a superior technique for exploring areas, where not much is known about what questions to ask or how to formulate questions.
- It offers a better opportunity than the questionnaire to evaluate the validity of the information gathered. The interviewer can observe not only what subjects say but also, how they say it.
- It is an effective technique for eliciting information about complex subjects and for probing the sentiments underlying expressed opinions.
- Many people enjoy being interviewed regardless of the subject. They usually cooperate in a study, when all they have to do is talk. In contrast, the percentage of returns to a questionnaire is relatively low, often less than 20 per cent. Attractively designed questionnaires that are simple to return, easy to follow and presented in a context that inspires cooperation improve the return rate.

The major drawback of interviews is the long preparation time. Interviews also take time to conduct, which means more time and money. So, whenever a more economic alternative captures the same information, the interview is generally not used.

Guidelines to a Successful Interview

Interview should be approached as logically as programming. In an interview, the following steps should be taken:

1. ***Stage Setting:*** This is an "ice-breaking", relaxed, informal phase, where the analyst opens the interview by focusing on the purpose of the interview, why the subject was selected and the confidential nature of the interview. The role of interviewee is like that of a reporter. He also has to evaluate the cooperation of the interviewer and adjust his/her own image to counter, the interviewee's. How the interview goes depends on, whether the interviewee is friendly, timid or resident expert type.
2. ***Establishing Rapport:*** In one respect, data collection is an imposition on user staff time and intrusion into their privacy. Even though, the procedure is authorized by the management in advance, many staff members are reluctant to participate. So, one must take care of the following:
   - Avoid misleading the user staff about purpose of study.
   - Assure confidentiality of interview.
   - Avoid personal involvement in affairs of user's department/work.
   - Avoid sharing information received from other sources.
   - Avoid acting like an expert or consultant.
   - Respect time schedules. Do not extend preoccupied time.
   - Do not make false promises.
   - Dress and behave properly.
   - Do not interrupt the talking.
3. ***Asking the Questions:*** Except in unstructured interviews, it is important that each question be asked exactly as it is worked. Ask questions in a predefined sequence.
4. ***Record the Response:*** Obtain the response in detail and note it down correctly.
5. ***Data Recording and Notebook:*** Care must be taken to record the data, their source, and time of collection. If there is no recording of conversation, the analyst runs the risk of not remembering enough details, attributing them to the wrong source, or otherwise distorting the data.

Questionnaires

In contrast to the interview is the questionnaire, which is a term used for almost any tool that has questions to which individuals respond. It is usually associated with self-administered tools with items of the closed or fixed alternative type. By its nature, a questionnaire offers the following advantages:

- It is economical and requires less skill to administer than the interview.
- Unlike the interviews, which generally question one subject at a time, a questionnaire can be administered to a large number of individuals simultaneously.
- The standardized wording and order of the questions and the standardized instructions for reporting responses ensure uniformity of questions. In contrast, the interview situation is rarely uniform from one interview to the next.

- The respondents feel greater confidence in the anonymity of a questionnaire than in that of an interview. In an interview, the analyst usually knows the user staff by name, job function, or other identification. With a questionnaire, respondents give opinion without fear that the answer will be connected to their names.
- The questionnaire places less pressure on subjects for immediate responses. Respondents have time to think the questions over and do calculations to provide more accurate data.

The advantages of self-administered questionnaire outweigh the disadvantages, especially when cost is a consideration. The principal disadvantage is a low percentage of return. Another disadvantage is that, many people have difficulty in expressing themselves in writing. Because of these disadvantages, interview is probably superior to questionnaire.

Types of Interviews and Questionnaires

Interviews and questionnaires vary widely in format and structure. Interviews range from the highly unstructured, where neither the questions nor the respective responses are specified in advance, to the highly structured alternative in which the questions and responses are fixed. Some variation within this range is possible.

The Unstructured Alternative

The unstructured interview is a relatively non-directive information gathering technique. It allows respondents to answer questions freely in their own words. The responses are spontaneous rather than forced. They are self-revealing and personal rather than general and superficial. The role of an analyst, as an interviewer is to encourage the respondent to talk freely and serve as a catalyst to the expression of feelings and opinions. This is best achieved in a permissive atmosphere in which the subjects have no feeling of disapproval.

The Structured Alternative

In the structured approach, questions are presented in exactly the same order to all subjects. If the analyst asks a subject, "Would you like to see a computerized approach to solving your accounts receivable problem"? and asks another subject "How do you feel about computers handling the accounts receivables"? the responses may not be the same, even though the subjects may both have the same opinion. Standardized questions improve the reliability of the responses by ensuring that all subjects are responding to the same questions.

Structured interviews and questionnaires may differ in amount of structuring of questions. Questions may be either closed or open-ended. An open-ended question requires no response direction or specific response. In a questionnaire, it is written with space provided for the response. Such questions are often used in interviews than in questionnaires because scoring takes time.

- Now that you have had the new installation for six months, how would you evaluate the benefits?
- What is your opinion regarding the "no smoking" policy in the DP centre?
- If you had a choice, how would you have designed the present information center?

Figure 4.1: Open-ended Questions

Closed Questions

Closed questions are those in which the responses are presented as a set of alternatives. There are five major varieties of closed questions.

1. Fill in the blank questions request specific information. These responses then can be statistically analyzed.

   Example

   - What is the name of MIS directory in your firm?
   - What is the average number of calls you receive from clients?

2. Dichotomous (Yes/No) type questions that offer two answers have advantages similar to those of the multiple choice type (explained later). The problem is making certain that, a reliable response can be answered by yes or no; otherwise, an additional choice (e.g., yes, no, don't know) should be included. The question sequence and content is also important.

   Example

   - Are you personally using a microcomputer in your business? (Please circle one)
   - Yes No
   - If not, do you plan to be using one in the next 12 months? (Please circle one)
   - Yes No
   - In the performance of your work, are you personally involved in the computer hardware/software purchase decisions? (Please circle one)

   Yes No

3. Ranking scale questions ask the respondents to rank a list of items in order of importance or performance. In the following example, the first question asks the respondent to rank five statements on the basis of, how they describe his/her present job.

   Example

   Please rank the five statements in each group on the basis of, how well they describe the job mentioned on the front page. Write "1" for the statement that best describes the job, write "2" for the next best and so on to "5" for the least well that describes the job.

   Workers on this job.

   .... are busy all the time.

   .... have work where they do things for other people.

   .... try out their own ideas.

   .... are paid well in comparison with other workers.

   .... have opportunities for advancement.

4. Multiple choice questions offer respondents specific answer choices. This offers the advantage of faster tabulation and less analyst bias due to the order in which the answers are given.

Respondents have a forward bias towards the first alternative item. Alternating the order in which answer choices are listed may reduce bias but at the expense of additional time to respond to the questionnaire. In any case, it is important to be aware of these types of biases when constructing multiple choice questions.

Example

What is the average salary of an entry level analyst? (Tick one)

.... under ₹ 15, 000

.... ₹ 15, 000 – ₹ 19, 999

.... ₹ 20, 000 – ₹ 24, 999

5. Rating scale questions are an extension of the multiple choice design. The respondent is offered a range of responses along a single dimension. In the following example, the respondent is asked to rate various aspects of his/her job on a scale of 1-5.

   Example

   How satisfied are you with the following aspects of your present job? (Please circle one for each question)

		Very Dissatisfied	Dissatisfied	No Opinion	Satisfied	Very Satisfied
1.	The way my job provides for steady employment	1	2	3	4	5
2.	The chance to be respon-sible for the work of others	1	2	3	4	5
3.	The pleasa-ntness of the working conditions	1	2	3	4	5

Figure 4.2

Advantages and drawbacks of the structured and unstructured interview and questionnaire techniques are listed below.

Interview Type	Advantages		Drawbacks	
Structured	1.	Easy to administer and evaluate due to standardization	1.	High initial preparation cost
	2.	Requires limited training	2.	Standardization of questions tends to reduce spontaneity
	3.	Easy to train new staff	3.	Mechanizes interviewing, which makes it impractical for all interview settings
Unstructured	1.	Provides for greater creativity and spontaneity in interviewing	1.	More information of use is gathered
	2.	Facilitates deeper understanding of the feelings and standing of the interviewee	2.	Takes more time to conduct, therefore, costly
	3.	Offers greater flexibility in conducting an overall interview	3.	Requires extensive training and experience for effective results.

Figure 4.3

Questionnaire Construction

The procedure for questionnaire construction consists of six steps.

1. Decide what data should be collected; define the problem to be investigated.
2. Decide on the type of questionnaire and then write the questions.
3. Outline the topics for questionnaire and then write the questions.
4. Edit the questionnaire for technical defects or biases that reflect personal values.
5. Pretest (try out) questionnaire to see how well it works.
6. Do a final editing to ensure that the questionnaire is ready for administration. This includes a close look at the content. Form a sequence of questions as well as the appearance and clarity of the procedure for using the questionnaire.

A critical aspect of questionnaire construction is the formulation of reliable and valid questions. To do a satisfactory job, the analyst must focus on question content, wording and format. The following is a checklist of what to consider.

- ***Question Content***
  - ❖ Is the question necessary? Is it a part of other questions?
  - ❖ Does the question cover the area adequately?
  - ❖ Does the subjects have proper information to answer the question?
  - ❖ Is the question biased in a given direction?
  - ❖ Is the question likely to generate emotional feelings that might lead to untrue responses?
- ***Question Wording***
  - ❖ Is the question worded for the subjects' background and experience?

- ❖ Can the question be misinterpreted? What else could it mean to a respondent?
- ❖ Is the frame of reference uniform for all respondents?
- ❖ Is the wording biased towards a specific answer?
- ❖ How clear and direct is the question?

- ***Question Format***
  - ❖ Can the question best be asked in the form of check answer or with a follow up free answer?
  - ❖ Is the response form easy to use or adequate for the job?
  - ❖ Is the answer to the question likely to be influenced by the preceding question? That is, is there any contamination effect?

Prototyping

Prototyping is the process of creating, developing and refining a working model of the final operational system. The prototype is a live working system and not just a paper-based design. Users can test its operations and explore its facilities and so, do not have to rely upon written descriptions.

Prototyping is an engineering discipline that has found its way into computer systems development. The idea, at least in the engineering parlance is to build small-scale working models of a product or its components. When it applies to computer systems development, it means that the analyst builds a small scale working model of the system or a sub-system.

Prototyping stresses the early delivery of a working system that could be used to refine user's requirements and system characteristics. Prototyping addresses the inability of many users to specify their information needs, and the difficulty of systems analysts to understand the user's environment by providing the user with a tentative system for experimental purposes at the earliest possible time. The user's experience with this experimental system leads to suggestions and tips for modifications to be incorporated into the prototype, thus leading to a new series of user experimentation.

Hence, the prototype is successively refined in a series of iterations until it eventually becomes an acceptable reflection of the user's requirements.

If prototyping is to be successful, then it is essential that the prototype is created quickly to give timely feedback to the development life cycle.

4.5 TECHNIQUES OF STRUCTURED ANALYSIS

To facilitate structured analysis a number of tools have been developed. These tools allow the analyst to represent and analyze the problem at hand in a systematic and logical manner. Earlier, system analysis heavily depended on the skill base of the analyst. As a consequence, analysts employed ad hoc tools suited to their individual tests and therefore, no standard were followed. With the development of the analysis aides that you will study in this section, analysis and modeling of the system have been not only standardized but also greatly simplified. Some of the frequently employed analysis tools are described hereunder.

4.5.1 Data Flow Diagrams

A DFD, also known as a "bubble chart", serves the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design phase that functionally decomposes the requirement specifications down to the lowest level of detail. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformations and lines represent data flows in the system. A basic DFD format is shown in Figure 4.4. The DFD is a representation of various processes and the input and output in each process. Further it also represents the various data stores.

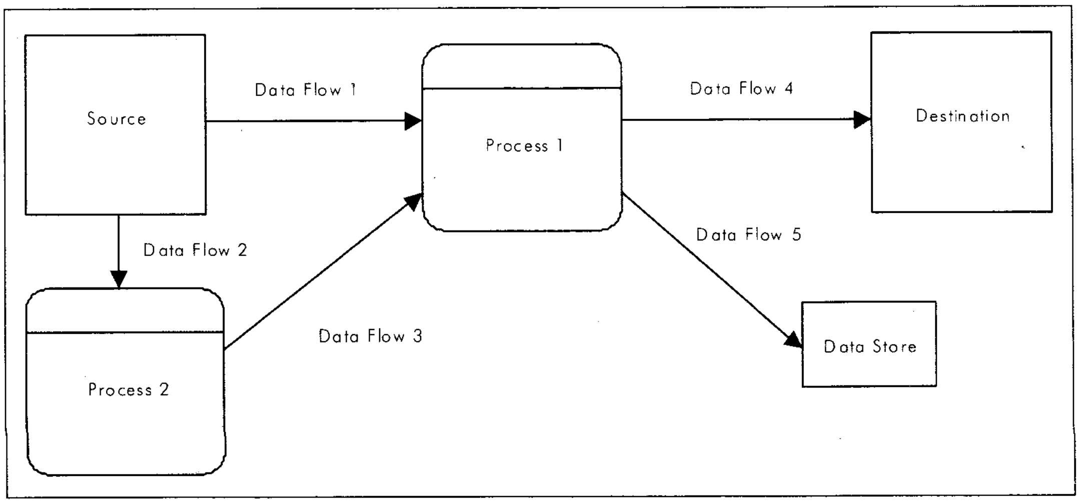


Figure 4.4: Sample DFD

Graphical description of a system's data and how the processes transform the data is known as Data Flow Diagram (DFD).

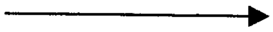
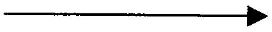
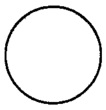
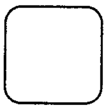
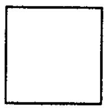
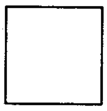
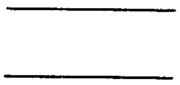
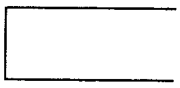
Unlike detail flow charts, DFDs do not supply detailed descriptions of modules but geographically, describe a system's data and how the data interact with the system.

Symbols Used in DFDs

- A square defines a source or a destination of system data. (External entity).
- An arrow identifies data flow - data in motion. It is a pipeline through which information flows.
- A circle or a bubble (some people use an oval bubble) represents process that transforms incoming data flow(s) into outgoing data flow(s).
- An open rectangle is a data store - data at rest - or a temporary repository of data.

Note that, a DFD describes data flow rather than how they are processed, so it does not depend on hardware, software data structure or file organization. The key question that we are trying to answer is: what major transformations must occur for input to be correctly transformed into output?

Several different notations or symbols have been suggested by different people. The most important are the notations suggested by Yourdon and by Gane and Sarson. The symbols used are:

	Yourdon	Gane and Sarson
1.	Data flow 	
2.	Process 	
3.	Source or Destination 	
4.	Data Store 	

Naming Conventions in DFD

Each component in a data flow diagram is labelled with a descriptive name. Process names are further identified with a number that will be used for identification purposes. The number assigned to a specific process does not represent the sequence of processes. It is strictly for identification and will take on added value when we study the components that make up a specific process.

The name of data stores, sources and destination are written in capital letters. Process and data flow names have the first letter of each work capitalized.

The names of the processes must clearly explain the actual task being performed in the process.

Developing a DFD

To be useful and informative, DFD must be drawn properly. This section shows, how to draw them: where to start, how to add details to description, how to be consistent in naming items included in the diagrams. The discussion also points out common mistakes that should be avoided.

Development Process

System analysts must first study the current systems, that is, the actual activities and processes that occur. In the terminology of structured analysis, this is the study of physical system. The physical system is translated into logical descriptions that focus on data and processes. It is advantageous to emphasize data and processes in order to focus on actual activities that occur and the resources needed to perform them, rather than on who performs the work.

These are examples of physical system details

Department	Copy room or building location
Person	Step number
File	Procedure

During data flow analysis, these details are evaluated in terms of the logical components of data flows, processes, data stores, origins and destinations.

During the design stages that follow, system requirements are translated into logical design details. Actual construction, such as the programming of computer software, translates logical specifications into physical features and a working information system.

This overview of the sequence of activities for analysis and design of information systems will be a backdrop for this discussion of data flow analysis.

Context Diagram

The context diagram is the top level DFD. It represents a system as a single process with only major inputs and outputs to the system represented as data flows. The source and destination of the data as external entities are also represented. Figure 4.5 represents a context diagram. This is a level O DFD for an accounts payable system. Note that, only a single process is shown. The vendor is both the source and the destination of data.

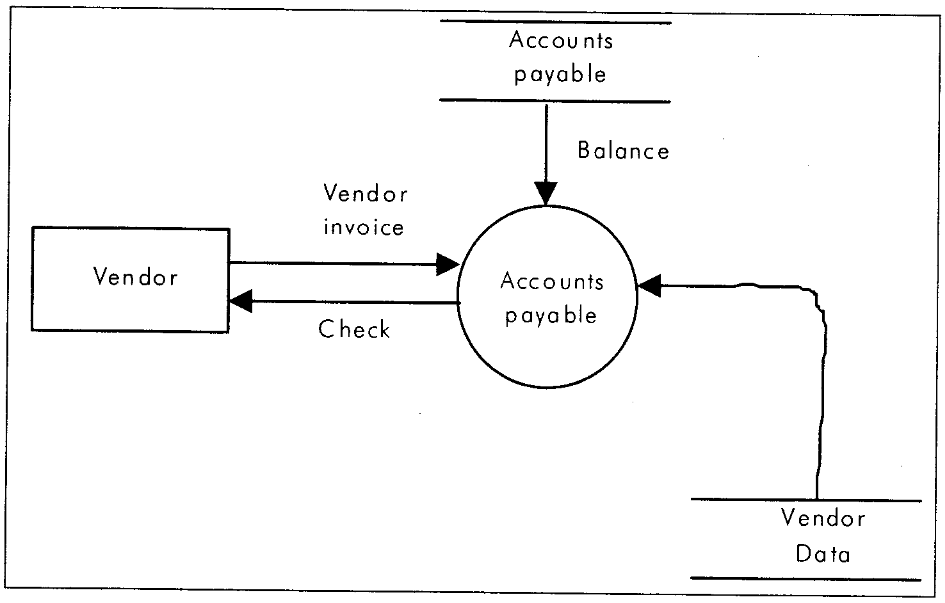


Figure 4.5: Context Diagram for Accounts Payable System

Level 1 and Higher Level DFD

When we explode the context diagram and break the single process into a number of detailed processes, then level 1 DFD is created. Further breaking of each process leads to further levels of DFDs. The numbering convention in the DFD is as follows:

- In the Context diagram, no number is given to processes as only one process exists.
- In the level 1 DFD, the processes are numbered as 1, 2, 3.
- In the next level, the processes are numbered depending on which process has been broken. For example, if process 2 of level 1 DFD is broken, the processes are numbered as 2.1, 2.2 and so on.
- The processes are numbered as 3.2.1, 2.3.2.1 and so on as further levels of DFD are created.

The rules of thumb in drawing DFDs are:

1. Process should be named and numbered for easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flows from the source (upper left corner) to the destination (lower right corner), although they may flow back to a source. An alternative way is to repeat the source symbol as a destination. Since, it is used more than once in the DFD, it is marked with a short diagonal in lower right corner.
3. When the process is exploded into lower level details, they are numbered. The sub-levels are numbered on the basis of their parent process number.
4. The names of data stores, sources, destinations must be in capital letters while the names of processes and data flows must start with a capital letter.

The rules stated above are not very hard and fast. However, it is a good practice to follow them.

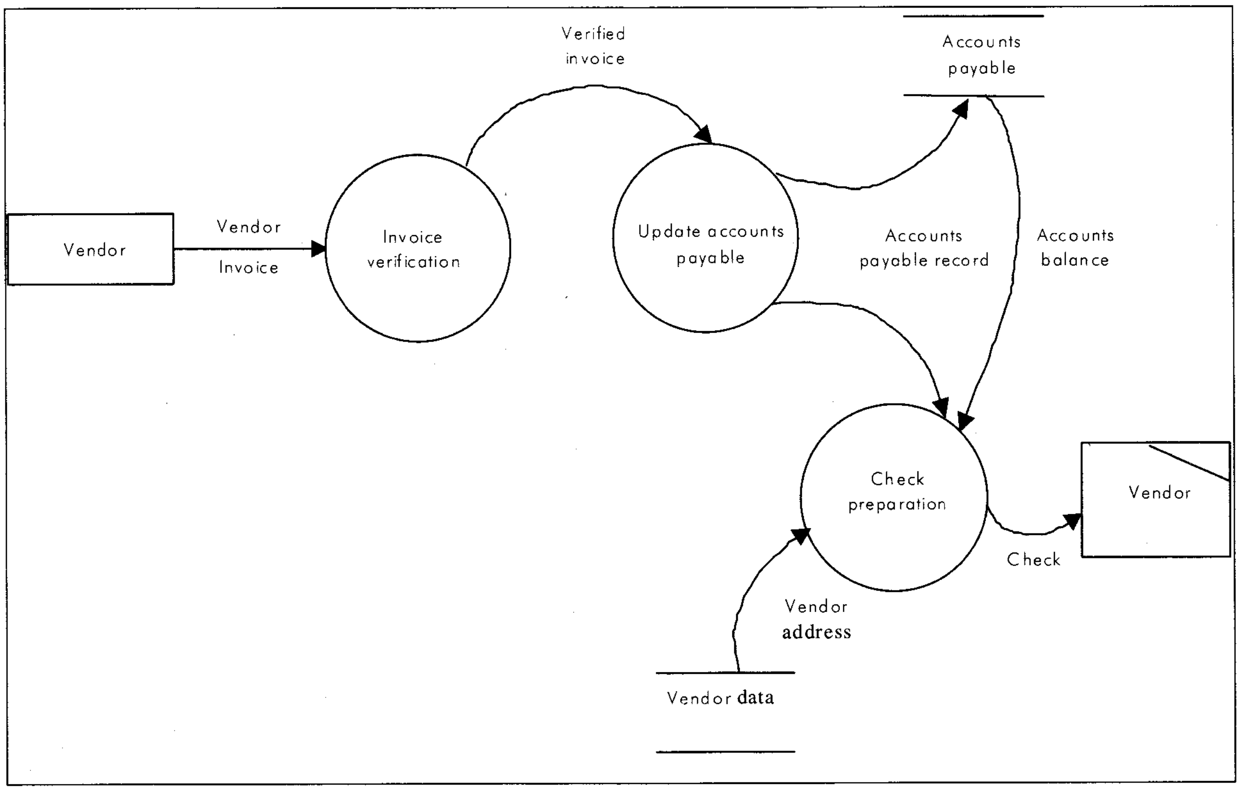


Figure 4.6: DFD for Accounts Payable System

4.5.2 Entity Relationship Diagram (ERD)

A system invariably deals with data. Data is similar type of meaningful facts or figures. In other words, data is the known facts that can be recorded and that have implicit meaning. It can also be defined as a representation of facts, concepts or instructions in a formal manner which is suitable for understating and processing by them or electronic

Representation of Data

Alphabets	(A-Z, a-z)
Digits	(0-9)
Special Characters	(+, ', -, *, #, @ etc.)

Record

Record is a collection of related data e.g., in file EMPLOYEE record for any employee is S.No. Employee Code, Employee Name, Date of Birth, Address, Basic, HRA, DA and Total Salary, These date items are also known as data fields.

Data Item

Data item or field is a set of characters, used together for representing a particular data element. In our example name of an employee is represented by the data item, Employee name.

Field Name

The name of each data item is known as the field name, e.g., Employee Code, Basic, HRA etc.

File

File is a collection of related records. File is of three types:

- File containing text is a text file.
- File containing program is a program file, and
- File containing data is a data file.

Here our consideration is data file so we are more focused on a data file concepts.

In our example, EMPLOYEE file might consists of the employee records for MANAV computers Pvt. Ltd.

Data and Information

Whereas data is some meaningful fact or figure, information is the processed data on which decisions and actions are based. Information can also be defined as the organized and classified data to provide the meaningful values to the receiver.

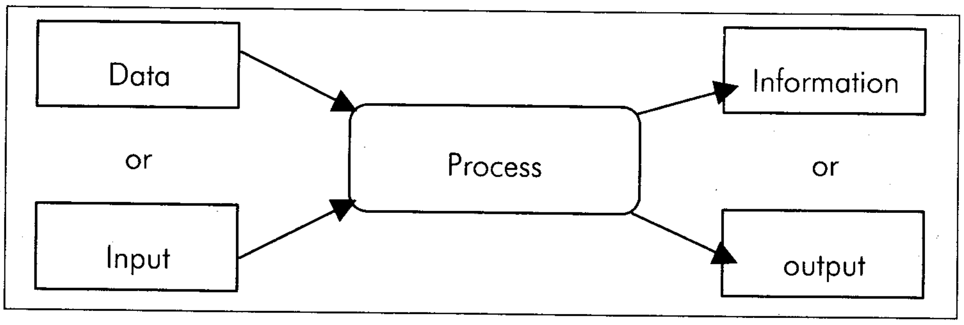


Figure 4.7

The ERD is graphical data modeling tool. It is based on a perception of a real worker that consists of a collection of basic objects, called entities, and of relationships among these objects. While DFD captures the process through which data gets transformed, ERD captures the data and its relationship.

ERD is built up by the following components:

- Rectangles, which represent entity sets
- Ellipses, which represent attributes
- Diamonds, which represent relationships among entity sets
- Lines, which link attributes to entity sets and entity sets to relationships.

For example, suppose we have two entities like customer and account, then these two entities can be modeled as follow:

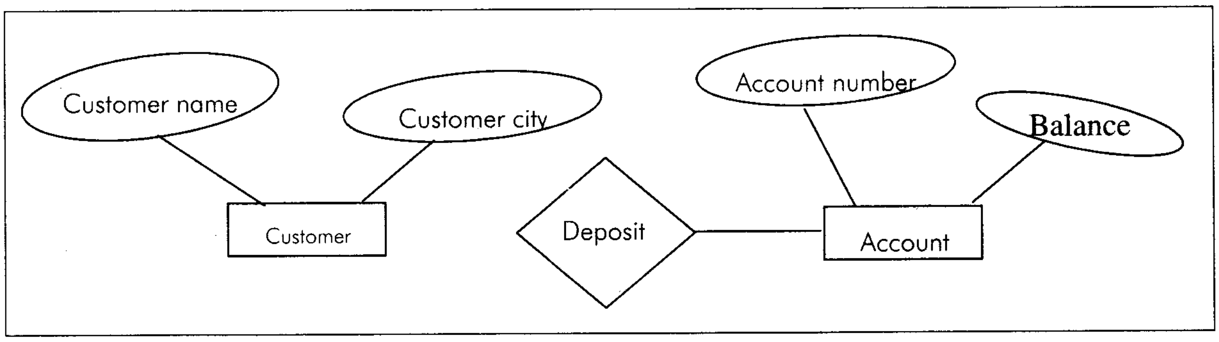


Figure 4.8: A Sample E-R Diagram

The entity-relationship model for data uses three features to describe data. These are:

1. Entities, which specify distinct real-world items in an application.
2. Relationships, which connect entities and represent meaningful dependencies between them.
3. Attributes, which specify properties of entities and relationships.

We illustrate these terms with an example. A vendor supplying items to a company, for example, is an entity. The item he supplies is another entity. A vendor supplying items are related in the sense that a vendor supplies an item. The act of supplying defines a relationship between a vendor and an item. An entity set is a collection of similar entities. We can thus define a vendor set and an item set. Each member of an entity set is described by some attributes. For example, a vendor may be described by the attributes:

(vendor code, vendor name, address)

An item may be described by the attributes:

(item code, item name)

Relationship also can be characterized by a number of attributes. We can think of the relationship as supply between vendor and item entities. The relationship supply can be described by the attributes: (order_no, date of supply)

ERD Notations

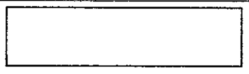
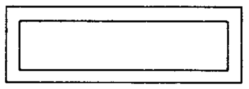
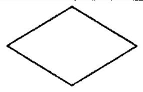
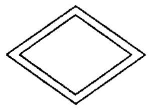
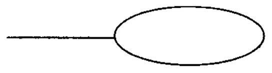
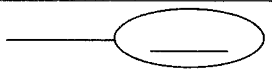
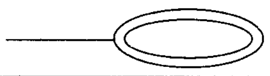
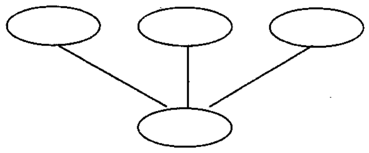
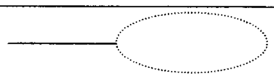
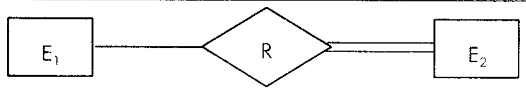
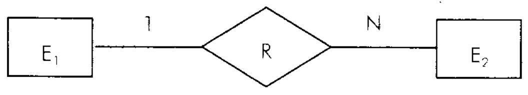
Symbol	Meaning
	Entity Type
	Weak Entity Type
	Relationship Type
	Identifying Relationship Type
	Attribute
	Key Attribute
	Multivalued Atrribute
	Composite Attribute
	Derived Atrribute
	Total Participation of E2 in R
	Cardinality Relation 1:N for E1:E2 in R

Figure 4.9

Relationship between Entity Sets

The relationship between entity sets may be many-to-many (M: N), one-to-many (1:M), many-to-one (M: 1) or one-to-one (1:1). The 1:1 relationship between entity sets E_1 and E_2 indicates that for each entity in either set there is at most one entity in the second set that is associated with it. The 1:M relationship from entity set E_1 to E_2 indicates that for an occurrence of the entity from the set E_1, there could be zero, one, or more entities from the entity set E_2 associated with it. Each entity in E_2 is associated with at most one entity in the entity set E_1. In the M: N relationship between entity sets E_1 and E_2, there is no restriction to the number of entities in one set associated with an entity in the other set. The data structure, employing the ERD model is usually shown pictorially using entity-relationship E-R diagram.

To illustrate these different types of relationships consider the following entity sets: Department, Manager, Employee, and Project.

The relationship between a Department and a Manager is usually one-to-one; there is only one manager per department and a manager manages only one department. This relationship between entities is shown in Figure 4.10. Each entity is represented by a rectangle and the relationship between them is indicated by a direct line. The relationship for Manager to Department and from Department to Manager is both 1:1. Note that a one-to-one relationship between two entity sets does not imply that for an occurrence of an entity from one set at any time there must be an occurrence of an entity in the other set. In the case of an organization, there could be times when a department is without a manager or when an employee who is classified as a manager may be without a department to manage.

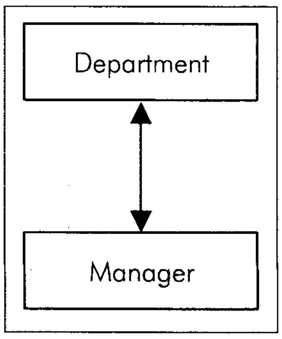


Figure 4.10: One-to-One Relationship

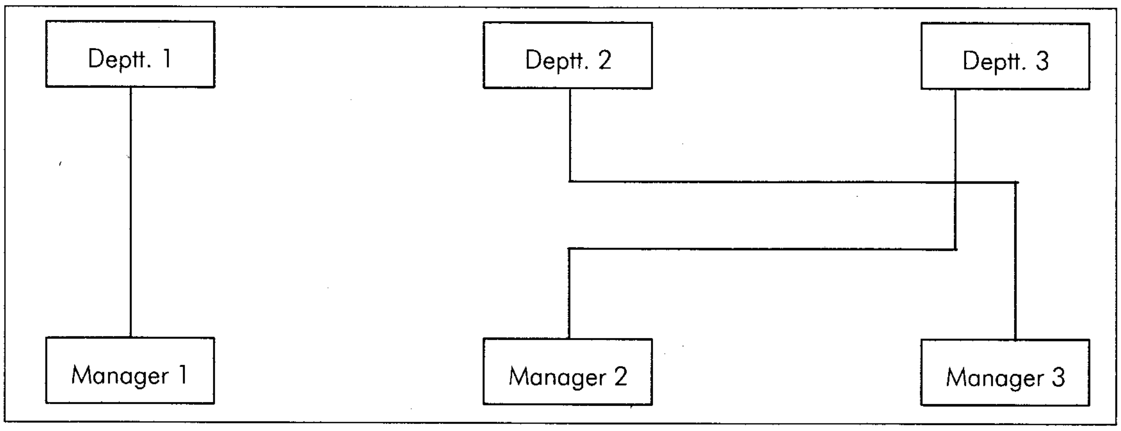


Figure 4.11: Some Instances of One-to-One Relationship

A one-to-many relationship exists from the entity Manager to the entity Employee because there are several employees reporting to the manager. As we just pointed out, there could be an occurrence of the entity type Manager having zero occurrences of the entity type Employee reporting to him or her. A reverse relationship, from Employee to Manager, would be many to one, since many employees may be supervised by a single manager. However, given an instance of the entity set Employee, there could be only one instance of the entity set Manager to whom that employee reports (assuming that no employee reports to more than one manager).

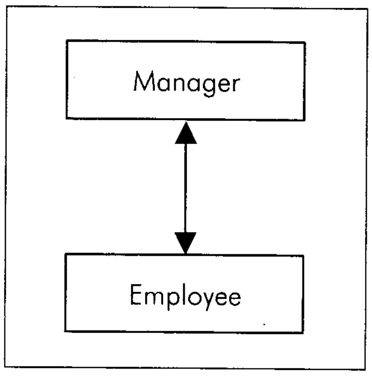


Figure 4.12: 1 : M Relationship

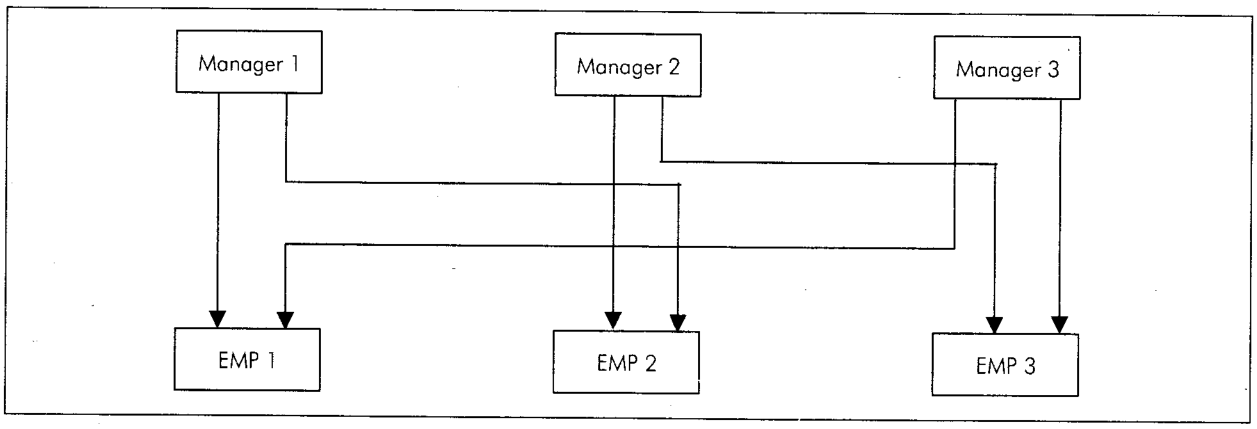


Figure 4.13: Instances of 1 : M Relationship

The relationship between the entity Employee and the entity Project can be derived as follows: Each employee could be involved in a number of different projects, and a number of employees could be working on a given project. This relationship between Employee and Project is many-to-many.

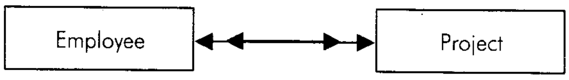


Figure 4.14 M:N Relationship

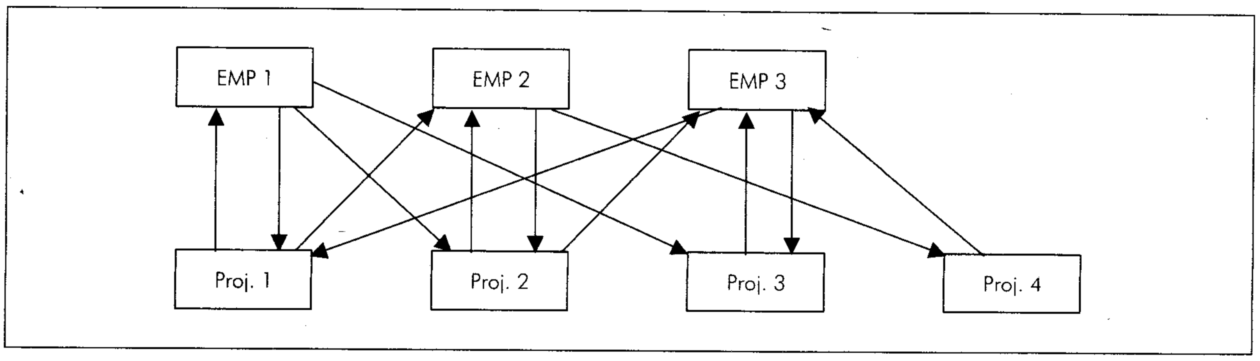


Figure 4.15: Instances of M : N Relationship

In the entity-relationship (E-R) diagram, entities are represented by rectangles, relationships by a diamond-shaped box and attributes by ellipses or ovals. The following E-R diagram for vendor, item and their relationship is illustrated in Figure 4.16.

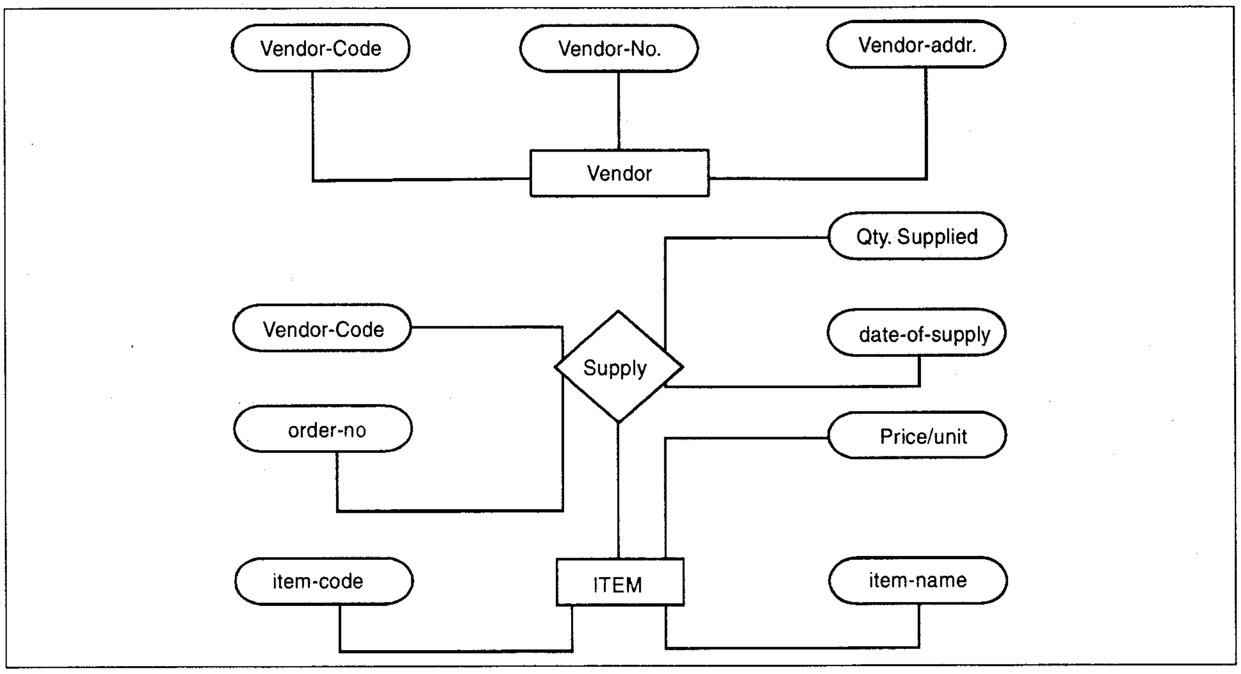
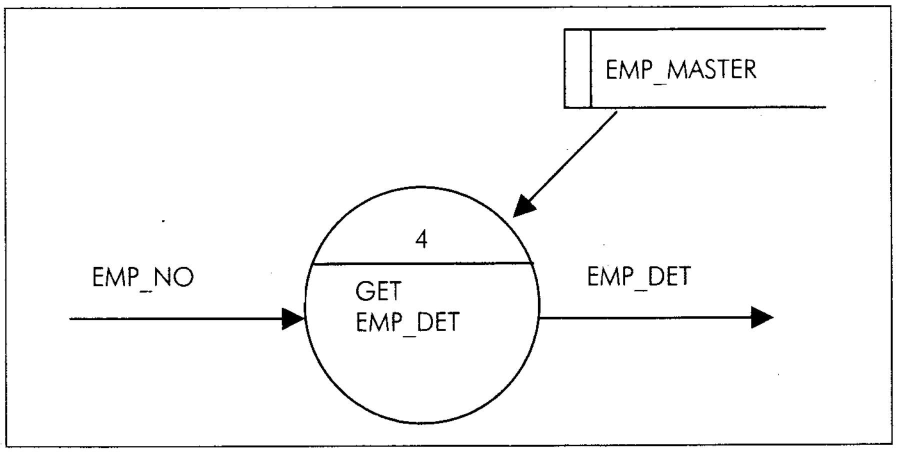


Figure 4.16: E-R Diagram for Vendor; Item and Their Relationship

4.5.3 Data Dictionary

A data dictionary is a catalog – a repository – of the elements in a system. As the name suggests, these elements center around data and the way they are structured to meet the user requirements and organization needs. In a data dictionary, you will find a list of all the elements composing the data flowing through a system. The major elements are data flows, data stores and processes. The data dictionary stores details and descriptions of these elements.

DD is an integral part of system specifications, since without it, DFDs are just pictures with no details, e.g.,



Data:

EMP_NO: Char (6)

Emp_details: EMP_NA+EMP_ADD+EMP_DOB

Data Stores:

EMP_RECORD: EMP_NO+EMP_NA+EMP_ADD+EMP_DOB

Process:

```
Do while.not.eof EMP_MASTER
open EMP_MASTER
go top
get EMP_DET
```

If analysts want to know, how many characters are in a data item, and by what other names it is referenced in the system, they should be able to find the answers in a properly developed data dictionary. Data dictionary is developed during data flow analysis and assists the analysts involved in determining system requirements.

Analyst uses data dictionary for five important reasons:

1. To manage the details in large systems.
2. To communicate a common meaning for all system elements.
3. To document the features of the system.
4. To facilitate analysis of the details in order to evaluate characteristics and determine where system changes should be made.
5. To locate errors and omissions in the system.

Components of a Data Dictionary

The dictionary contains two types of descriptions for the data flowing through the system: data elements and data structures. Data elements are grouped together to make up a data structure.

Data Element

The most fundamental data level is the data element. No smaller unit has meaning to the system analyst or the user. For example, invoice number, invoice date, etc. Data elements are building blocks for other data in a system. The data element entry consists of the following items:

- Data Name
- Data Description: Brief description of what the data element represents
- Aliases or additional names for the data element
- Length or size of the element
- Data Values: It is the list of possible values for the data element

Example

Data Name:	Part_Color
Data Description:	This is the color of the part of the machinery and is ordered time to time from local vendor
Aliases:	P_Colour, Color
Length:	Text of Length 15
Data Values:	May take one of the following values. Red, Orange, Yellow, Blue, Purple, Pink

Data Structure

A data structure is a set of data items that are related to one another and that collectively describe a component in a system. Both data flows and data stores are data structures.

Data structures are built on four relationships of components which may be data elements, or other data structures (see Figure 4.17).

(a) ***Sequence Relationship:*** Defines the components that are always included in a particular data structure; a concatenation of two or more data items.

(b) ***Selection (Either/Or) Relationship:*** Defines alternative data items included in a data structure.

(c) ***Iterational (Repetitive) Relationship:*** Defines the repetition of a component, zero or more times.

(d) ***Optional Relationship:*** A special case of iteration; data elements may or may not be treated zero or one time.

The description of the above relations can be found in Figure 4.17. Figure 4.18 shows an example of defining structure using these sets of relationships.

Symbol	Meaning	Explanation	Use
=	is equivalent to	Alias	Denotes synonyms.
+	and	Concatenation	Denotes sequence relationship. Defines components
	always included in a		
			particular data structure
[ ]	either/or	Defines alternate	Denotes selection relationship. components of data
	structure		
{ }	iterations of	Defines the repetition	Denotes iteration relationship. of a component in a
	data structure		
( )	optional	Defines iteration that	Denotes optional relationship. occurs only 0 or 1
	times		
Aliases		Another name	

Figure 4.17: Notation used to Show Structural Relationships in Data

Invoice	= Payment request
Payment voucher	= Invoice package + Payment approval
Invoice receipt	= Signed invoice
Audit approval	= Audited invoice
Purchase authorization	Purchase order number + Authorization date = Manager approval
Item details	= Item number + Item description + Item cost + Item extension
Amount of invoice	= {Item extension} + Shipping amount {+ Sales tax}
Vendor balance	= Beginning balance +{Purchases} + {Payments} + {Credits}
Beginning balance	= Vendor balance sets up next month cycle*
Symbols:	= Is equivalent to
	+ and
	[ ] either/or
	{ } iterations of
	( ) optional
	* encloses annotation

Figure 4.18 Data Descriptions and Notation

Two more terms need to be clarified for the purpose of understanding Data Dictionary:

Data Store:	Data store is a data structure for collecting data input during processing. For example, Part Register is a data store.
Data Flow:	Data flow is a data structure, that shows a unit of data in motion. For example, New Part Details is a data flow that moves from an external entity to a process.

Importance of Data Dictionary

Data dictionary is an important tool for structured analysis as it offers the following advantages:

- It is a valuable reference for designing the system. It is used to build the database and write programs during design phase.
- It assists in communicating meanings of different elements, terms and procedures.
- It facilitates analysis in determining additions and changes in the system.
- It helps the analyst to record the details of each element and data structure.
- It is used to locate errors in the system descriptions.
- It is also a useful reference document during implementation of the system.

Four Rules

Four rules govern the construction of data dictionary entries:

1. Words should be defined to stand for what they mean and not the variable names by which they may be described in the program; use CLIENT_NAME not ABCPQ or CODE06. Capitalization of words help them to stand out and may be of assistance.
2. Each word must be unique; we cannot have two definitions of the same client name.
3. Aliases, or synonyms, are allowed when two or more entries show the same meaning. A vendor number may also be called a customer number. However, aliases should be used only when absolutely necessary.
4. Self-defining words should not be decomposed. We can even decompose a dictionary definition. For instance, we might write

Vendor name	=	Company name, Individual's name

which we might further decompose to

Company name	=	(Contact) + Business name
Individual's name	=	Last name + First name + (Middle initial)

There are two kinds of data dictionaries:

(i) integrated and

(ii) stand-alone

The integrated dictionary is related to one database management system. To the extent the organization data is under this DBMS it is global or organization wide. However, very few enterprises have all their data eggs in one basket, so the dictionary documentation (metadata) can be considered as local and fragmented.

The stand-alone dictionary is not tied to any one DBMS, although it may have special advantages for one DBMS, such as the IBM DB-DC Data Dictionary, which has special features related to the IBM IMS DBMS, but is still a stand-alone variety of dictionary.

Both these types of dictionaries can be identified by functions as either passive, active, or inline. Viewed either way, by type or function, the differences are striking. Passive, active, and in-line dictionaries differ functionally as follows:

- *Passive Data Dictionaries:* The functionally passive dictionary performs documentation only. This variety of dictionary could be maintained as a manual rather than an automated database. For more than limited documentation use, the automated passive dictionary has clear advantages. From the organizational view, the documentation function is the most important dictionary service with the most potential benefits, so the passive dictionary should not be thought of negatively. It has more limited functionality but may perform its critical function of global documentation best of all.
- *Active Data Dictionaries:* Besides supporting documentation to one degree or another, the active data dictionary supports program and operations development by exporting database definitions and program data storage definitions for languages such as COBOL and Job Control Language (JCL) for execution-time performance. The IBM DB/DC Data Dictionary already mentioned is such a stand-alone, active data dictionary. A dictionary, such as this is not an in-line data

dictionary as delivered, which is not to say that it could not be put in-line by a determined effort of major proportions.

- ***In-line Data Dictionaries:*** An in-line data dictionary is active during program execution, performing such feats as transaction validation and editing. Such a dictionary would always have some documentation value, but documentation across the organization about the organization's functions and activities and all the organization's information data stores is not likely. In-line dictionaries are associated with DBMS products, such as Cullinet Software Corporation's IDMS-R or Cincom System's TOTAL, to name just two.

4.5.4 Decision Tree

Decision tree is a tool for documenting procedures where actions need to be taken depending on a number of conditions. If the group of conditions has a specific value, a certain action is taken. The action taken differs with the value of group of conditions. In such cases, decision trees are used.

A decision tree is a diagram that presents conditions and actions sequentially and thus shows which conditions are considered first, which second and so on. It is also a method of showing the relationship of each condition and its permissible actions. The diagram resembles branches on a tree and hence, the name (Figure 4.19).

The root of the tree, on the left of the diagram is the starting point of the decision sequence. The particular branch to be followed depends on the conditions that exist and the decision to be made. Progression from left to right along a particular branch is the result of making a series of decisions. Following each decision point is the next set of decisions to be considered. The nodes of the tree, thus, represent conditions and indicate that a determination must be made about which condition exists before the next path can be chosen. The right side of the tree lists the actions to be taken, depending on the sequence of conditions that is followed.

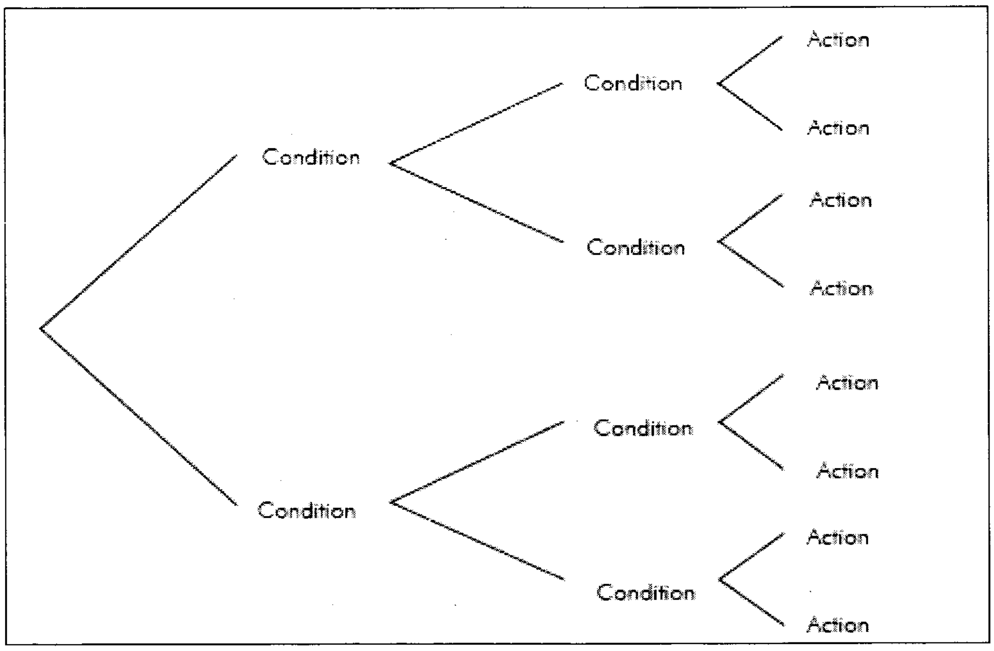


Figure 4.19 The Decision Tree Sequence

Using Decision Tree

Developing a decision tree is beneficial to an analyst in two ways. First of all, the need to describe conditions and actions forces the analyst to formally identify the actual decisions that must be made. It becomes difficult for them to overlook an integral step in the decision process, whether it depends on quantitative or non-quantitative variables.

It is possible, for instance, to show what discount action to take, depending on the number of dollars spent by customers. When an organization opens accounts with dealers and suppliers, it formalizes an agreement for taking discount from the full invoice price. Two conditions are specified in this agreement: first, the invoices must be payed within 10 days of receipt, and second, the size of discount will depend on the value of the invoice. It is agreed that under some conditions, the organization can take the action of deducting 3 per cent, under other conditions a 2 per cent discount and under all other conditions, no discount is offered.

Decision tree also forces analysts to consider the sequence of decisions. Consider the sequence of decisions in the example (Figure 4.20). You can quickly determine that one condition - the amount of. the invoice - does not matter until the condition of payment duration is met. Therefore, the decision tree identifies the time condition first and shows the two values (within 10 days and more than 10 days). The discount condition is described next, but only for the branch of the tree 'within ten days'. The more than 10 days' has no other relevant condition and so it shows the relevant action (unconditionally). The tree shows that action "Pay full invoice amount" applies under two different conditions. It also shows that, there is no reason to pay invoice of less than ₹ 5000 within 10 days, since there is no discount available for these amounts.

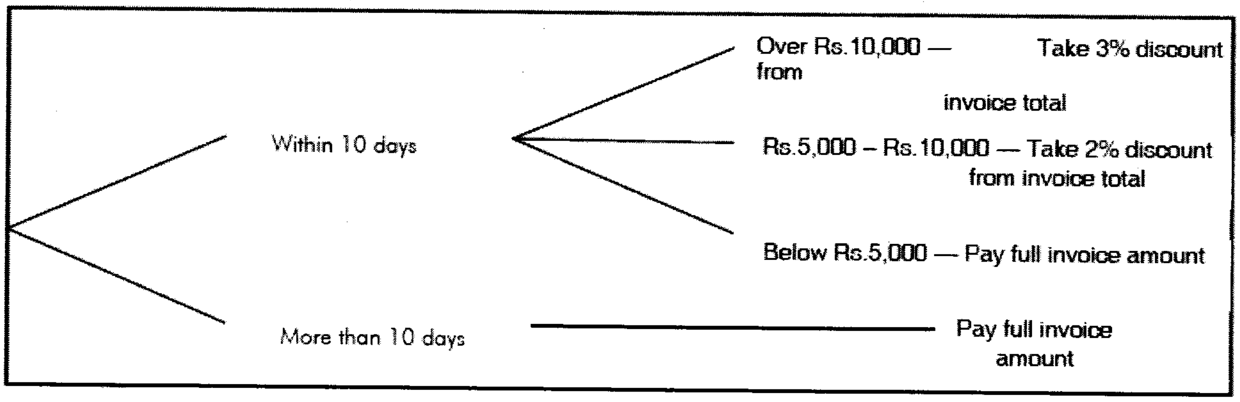


Figure 4.20: Decision Tree for Discount Authorization

Another example shows, a none-quantitative condition for processing accounts payable: signed invoices, authorize purchase and correcting price. Depending on the three sets of conditions that exist, one of two actions can be taken - payment can be authorized or the submitted invoice can be rejected. Notice how clearly each alternative is shown in the decision tree. Sometimes, in more complex business situations, the specific action, most appropriate under several conditions is not readily apparent with formal analysis of this nature.

The sequence of decisions is easy to see in the example. The condition of having a valid purchase order does not matter unless the invoice has been signed. The signature is important because the condition of having a signed invoice must be met before processing can continue. The analyst can consider the authorization condition.

Decision trees may not always be the best tool for decision analysis. A decision tree for a complex system with many sequences of steps and combinations of conditions may be unwieldy. A large number of branches, with many paths through them, will cloud rather than aid analysis. The analyst may not be able to determine, which business policies or practices guide the formulation of specific decisions. Where these problems arise, decision tables should be considered.

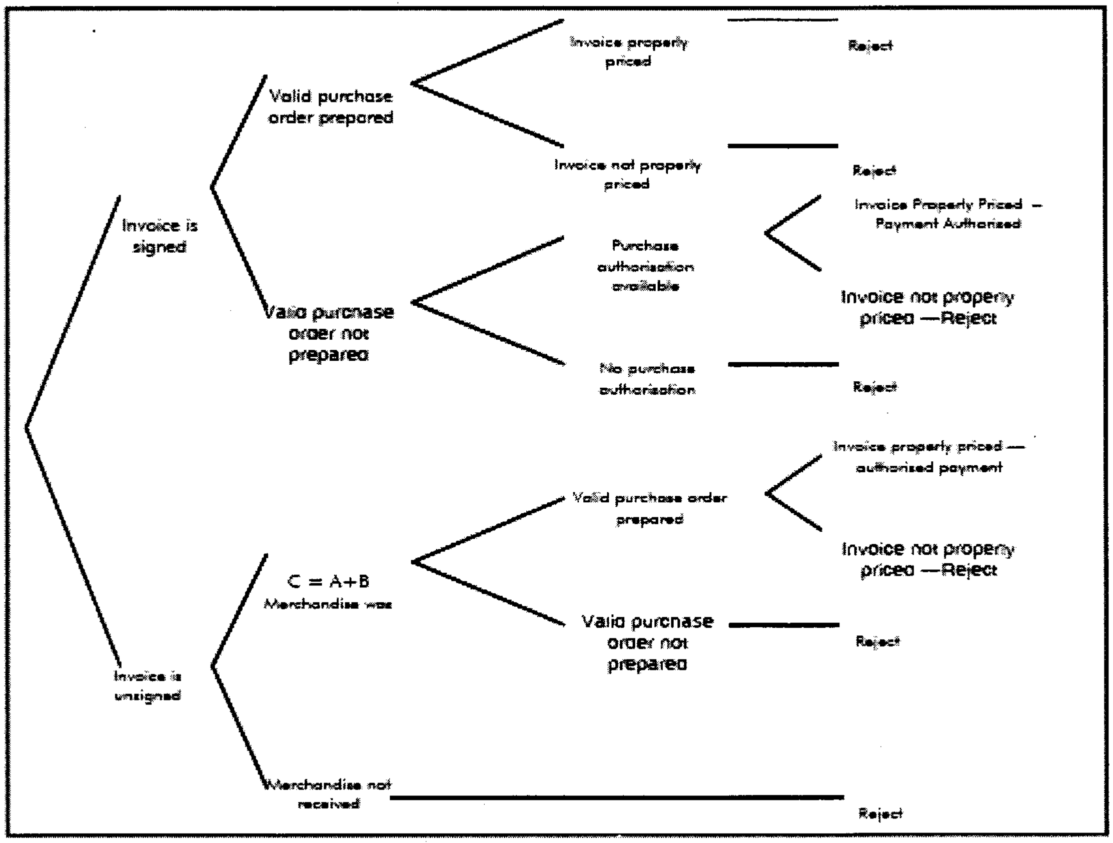

Invoice is signed
Valid purchase order prepared
Invoice properly priced
Reject
Invoice not properly priced
Reject
Valid purchase order not prepared
Purchase authorisation available
Invoice Properly Priced – Payment Authorised
Invoice not properly priced —Reject
No purchase authorisation
Reject
Invoice is unsigned
C = A+B
Merchandise was
Valid purchase order prepared
Invoice properly priced — authorised payment
Invoice not properly priced —Reject
Valid purchase order not prepared
Reject
Merchandise not received
Reject

Figure 4.21: Decision Tree for Invoice Processing Example

Decision Tables

A decision table is a matrix of rows and columns, rather than a tree, that shows conditions and actions. Decision rules, included in a decision table, state what procedures to follow when certain conditions exist. This method has been used in analysis of business functions, such as inventory control, sales analysis, credit analysis and transportation control and routing.

Characteristics of a Decision Table

The decision table is made up of four sections: condition statement, condition entries, action statements and action entries (Figure 4.22). The condition statements identify the relevant conditions. Condition entries tell which value, if any, applies for a particular condition. Action statements list the set of all steps that can be taken when a certain condition occurs. Action entries show, what specific action in the set to take when selected conditions or group of conditions are true. Sometimes notes are added below the table to indicate when to use the table or to distinguish it from the other decision tables.

CONDITIONS	DECISION	RULES
Condition	Condition	Entries
Action Statement	Action	Entries

Figure 4.22: General Form of a Decision Table

The column on the right side of the table, linking conditions and actions, form decision rules which state the conditions that must be satisfied for a particular set of actions to be taken. Notice that, we have dropped the ordering sequence that was necessary with decision trees. The decision rule incorporates all the conditions that must be true, not just one condition at a time.

The decision table in Figure 4.23 describes the actions taken in handling payments from patients in a medical clinic. The actions taken depend on, whether the patient has medical insurance and if so, which type. Two type of insurance coverage are identified: basic health insurance and social health insurance. The existence of a kind of insurance is stated by Y or N (for yes and no respectively). Four rules relate the combination of conditions 1 and 2 to three different actions. The patient must pay the cost of the office call, but no other charges; the patient pays none of the charges; or the patient pays full cost of the treatment.

Condition	**Decision Rules**			
	1	2	3	4
C1. Patient has basic health insurance	Y	N	Y	N
C2. Patient has social health insurance	N	Y	Y	N
Actions				
A1. Pay amount of office call	X			
A2. Pay nothing		X	X	
A3. Pay full amount of service				X

Figure 4.23: Clinic Payment Decision Table

From reading the table in Figure 4.23, it is clear that, when conditions C1 and C2 are yes and no respectively, the action A1 is carried out. Similar inference may be made on other three rules.

The discount on invoice problem discussed and solved through decision tree in Figure 4.20 is represented as decision table in Figure 4.24. The Y/N format (Y and N) is used in the condition entry section but actual values for each of the conditions are shown in the condition statement section. The format does not change the usefulness of decision table.

Building Decision Tables

To develop decision tables, analysts should use the following steps:

- Determine the most relevant factors to be considered in making a decision. This identifies the conditions in the decision. Each condition selected should have the potential to either occur or not occur, partial occurrence is not possible.
- Determine the most feasible steps or activities under varying conditions. This identifies the actions.
- Study the combinations of conditions that are possible. For every number N of conditions, there are 2N combinations to be considered. For example, for three conditions, there are eight possible combinations; 23 = 8. For four, 24 = 16 combinations are possible and can be included in the table.
- Fill in the table with decision rules. There are two ways to fill in the table.

  The first and longer method is to fill in condition rows with a yes or no value for each possible combination. That is, fill the first half of rows with Y and other half with N. Then follow with 25%Y, 25%N, 25%Y, 25%N, then fill the next row with 12.5% of Y and N alternatively and so on.

  The other method deals with one condition at a time and does not add duplicate combinations of conditions and actions, as discussed below.

  - State the first condition and permissible actions.
  - Add the second condition by duplicating the first half of the matrix and filling in different Y and N values from the new condition in both halves of the expanded matrix.
  - Repeat previous step for each additional condition.
- Mark action entries with X to signal action(s) to be taken; leave cells blank or mark with a dash to show that no action applies to that row.
- Examine the table for redundant rules or for contradictions within rules (discussed below).

Condition	Decision Rules			
	1	2	3	4
C1. Patient has basic health insurance	Y	N	Y	N
C2. Patient has social health insurance	N	Y	Y	N
Actions				
A1. Pay amount of office call	X			
A2. Pay nothing		X	X	
A3. Pay full amount of service				X

Figure 4.24: Payment Discount Example

Condition	Decision Rules					
	1	2	3	4	5	6
Time within	Within 10 days	Within 10 days	Within 10 days	Not within 10 days	Not within 10 days	Not 10 days
Business Volume	Over Rs.10,000 Rs.5,000	Rs. 5,000 to Rs.10,000	Below Rs.5,000	Over Rs.10,000	Rs.5,000 to Rs.10,000	Below
Take 3% discount	X					
Take 2% discount		X				
Pay full invoice amount			X	X	X	X

Figure 4.25: Decision Table for above Problem using Yes/No Format

Checking Decision Tables

After constructing a table, analysts verify it for correctness and completeness to ensure that, the table includes all the conditions, along with decision rules that relate them with actions. Redundancy and contradictions must also be removed.

Eliminate Redundancy: Decision tables can become large and unwieldy, if allowed to grow in an uncontrolled fashion. Removing redundancy can help reduce the size of the table. Redundancy occurs when both of the following are true.

- Two decision rules are identical except for one condition row, and
- The actions for the two rules are identical.

Remove Contradictions: Decision rules contradict each other when two or more rules have the same set of conditions and the actions are different. (Remember, if they are the same, then the decision rules are redundant).

Contradictions either mean that the analyst's information is incorrect or that, there has been an error in the construction of the table.

Unnecessary rules can be avoided. For example, in the case of invoice discount, look at Figure 4.26, the first three columns have first condition as N and the result always comes to be "pay full amount" because when the payment period is not within 10 days, we do not need to ask any further questions. The first three columns may be converted into one column. Similarly, we can reduce the rules, by not asking unnecessary conditions. An example of reducing the size is given in Figure 4.26.

Condition	Decision Rules			
Within 10 days	Y	Y	Y	N
Over Rs.10,000	Y	N	N	-
Rs.5,000 - Rs.10,000	-	Y	N	-
Below Rs.5,000	-	-	Y	-
3% discount	X			
2% discount		X		
Full amount			X	X

Figure 4.26

Types of Table Entries

- *Limited Entry Form:* The basic table structure used in the previous examples, consisting only of Y, N, and blank entries is a limited entry form. It is one of the most commonly used formats. Two other forms are also widely used.
- *Extended Entry Form:* This replaces Y and N with action entries, telling the reader how to decide. In this format, the condition and action statements themselves are not complete, which is why the entries contain more details than Y and N.
- *Mixed Entry Form:* Analysts may prefer to combine features of both the limited and extended entry forms, in the same table. Generally, only one form should be used in one section of the table but between the condition and action section, either form can be used. The example in Figure 3.16 presents condition entry in extended form and action entry in limited form.

4.5.5 Structured English

Structured English uses narrative statements to describe a procedure. It uses three basic types of statements:

(a) ***Sequence Structures:*** They include a set of instructions that are carried out one after another and do not depend on any condition.

(b) ***Decision Structures:*** They include one or more sets of instructions that are carried out depending upon one or more conditions. They generally use the phrase IF THEN ELSE to carry out different actions.

(c) ***Iteration Structures:*** They include a set of instructions that are repeated until a particular condition occurs. They generally use the phrase DO WHILE ...ENDDO to repeat a set of instructions.

The examples of these three types of statements are illustrated in Table 4.1.

Table 4.1: Examples of Three Types of Statements

Sequential Structure	Decision Structure	Iteration Structure
Accept employee code Accept employee name Accept other details Store data	If Basic_Pay < = 1000 HRA = 500 Else If Basic_Pay < = 3000 HRA = 1000 Else HRA = 1500 Endif Endif	Ans = "Y" Do while Ans = "Y" Accept employee code Accept employee name Accpet other details Display "Continue (Y/N)?" Accept Ans Enddo

4.5.6 Systems Documentation

A system cannot be completely effective unless it is adequately documented. It should be documented as it is being created. That is, at various stages of the system development, status reports should be prepared for those management personnel for whom the system is being designed. Such reports could include flowcharts, decision tables, output forms and other documents so far developed. It also includes various problems encountered, suggested solutions and resulting schedule revisions. In this way, management remains fully aware of systems' progress so that they can offer criticisms or suggest changes, while it is still economically and physically possible to make these changes without it being necessary to revise the entire system. These progress reports provide an excellent basis on which to build additional documentation.

Definition of Documentation

After successful testing of the system, all the work done during systems analysis and design is required to be properly organized. The organized way of keeping records of all the documents, programs and diagrams prepared during all the phases of system development life cycle (SDLC) is called documentation. All types of written instructions, which are prepared during SDLC and are required for operating and maintaining the system, must be included in the documentation.

Types of Documentation

These are five major types of documentation. They are:

(i) Program Documentation

(ii) Operations Documentation

(iii) User Documentation

(iv) Management Documentation

(v) System Documentation.

(i) *Program Documentation:* Many companies discuss about programming documentation but fail to provide it adequately. Before a program is developed, the systems analyst should provide the programmer with the required documentation. The logic in some programs is best described by a flowchart. Sometimes, decision tables are most appropriate for explaining the logic of a program. Programmers should insist on proper documentation before starting a job.

(ii) *Operations Documentation:* A well designed system may run for a long time with little or no assistance from the systems department. This can happen only when the system has been documented in a proper way. For smooth running of the system, the console operator must have complete knowledge about the job. Providing the computer centre with a set of operating instructions will not serve the purpose. The instructions must be in a form readily accessible to the console operator and written in simple and understandable style. A systems analyst must thoroughly discuss all the requirements of new jobs with the operations staff before the job can be properly transferred.

(iii) *User Documentation:* Systems users require proper documentation to prepare a developing system and to smoothly carry out existing ones. To meet this requirement, each system should have a manual that spells everything that the users must know to do their job correctly. Users require two general types of information; complete details to handle each case of the system processes, and overall picture of the system so that they can see their role in the total operation of the company.

(iv) *Management Documentation:* The documentation required by corporate management differs quite a lot from that required by users. The systems designer must know the requirements of the management and provide documentation to enable management to perform three functions:

   (i) Evaluate progress on systems development

   (ii) Monitor existing systems

   (iii) Understand the objectives and methods of new and existing systems

   Management is primarily interested to know, in general, the systems' overall objectives and basic operations. A brief manual highlighting the key steps in each system may be prepared for management. Good managers have an exceptional ability to get to the root of a system and their experience should enable them to retrieve information from a systems summary or chart which may not be apparent to the systems analyst.

(v) *System Documentation:* Each phase in the system development cycle is accompanied by appropriate documentation. The systems request, even if it is initially merit, verbally, eventually, must be written. It is desirable for the client and a systems analyst to work jointly in writing the request since each can contribute knowledge the other does not have. The written systems' request is merely a statement of the user's problem.

Documentation also includes plans to test the system and convert from the old to the new one. The systems analyst must also provide a plan to train the personnel affected by the changes.

During the life cycle of the completed system, the system itself must provide documentation of, how well it is operating and consequently should be designed to yield data about itself as a normal by-product.

Documentation Tools

A system is required to meet the needs of maintenance, reliability and testing. For this, you need well-designed modular software. The following three tools can be used to measure the affectivity of a system:

- Structured flowchart
- HIPO diagrams
- Wanier/Orr diagrams

Structured Flowchart

A structured flowchart is a graphic tool that forces the system designer to structure the software in modular as well as top-down form. It is also called Nassi-Schneidermann chart. The main advantage of using the structured flowchart is that it provides a proper structure that can be retained by the programmer for developing the application software. However, to ensure effective use of structured flowcharts, a programmer should be an expert in using these.

There are three basic elements or symbols that are used in developing structured flowcharts. They are being listed below.

- Decision symbols
- Process symbols
- Iteration symbols.

Decision Symbols

The decision symbol symbolizes alternative conditions that can occur. The program must have a manner of handling these alternative conditions. The decision symbol may show actions for more than two alternatives at the same time.

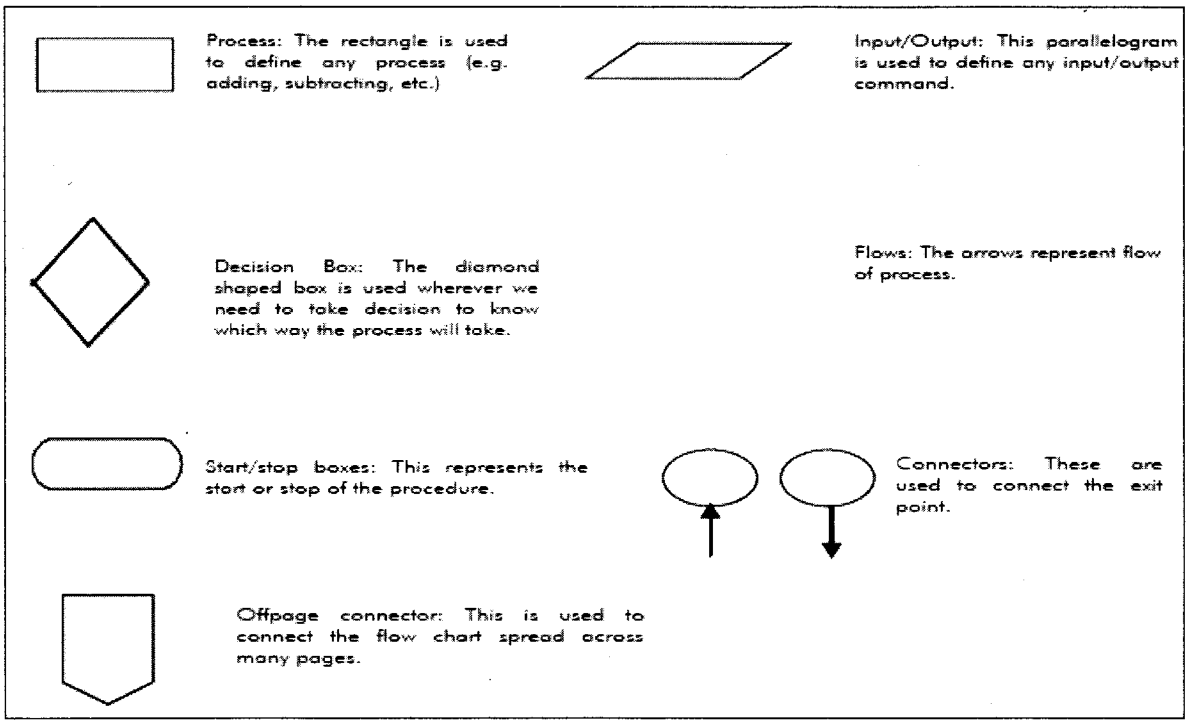
Process Symbols

The process symbols are rectangular boxes that represent a simple process or steps in a program. The process symbol represents initialization of values, input and output operations and calls to execute other procedures.

Iteration Symbols

Looping and repetition of operations is represented by the iteration symbols. These represent looping while a certain condition exists or until a condition exists.

Each structured flowchart is ideally displayed on a single sheet of paper. It uses no arrows or continuations on separate pages. The logic in a structured flowchart is shown in a top-down fashion. This feature constitutes an essential element of a structured flowchart. The first consideration in a process is the top element. The second in sequence is next one shown and so forth. Similarly, there is a single exit from the process.


Process: The rectangle is used to define any process (e.g. adding, subtracting, etc.)
Input/Output: This parallelogram is used to define any input/output command.
Decision Box: The diamond shaped box is used wherever we need to take decision to know which way the process will take.
Flows: The arrows represent flow of process.
Start/stop boxes: This represents the start or stop of the procedure.
Connectors: These are used to connect the exit point.
Offpage connector: This is used to connect the flow chart spread across many pages.

An example of flow chart in Figure 4.27 represents the flow of logic while adding two numbers.

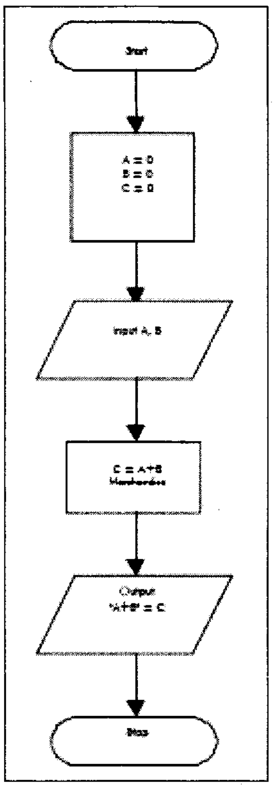


Figure 4.27: Programme Flow Chart - An Example

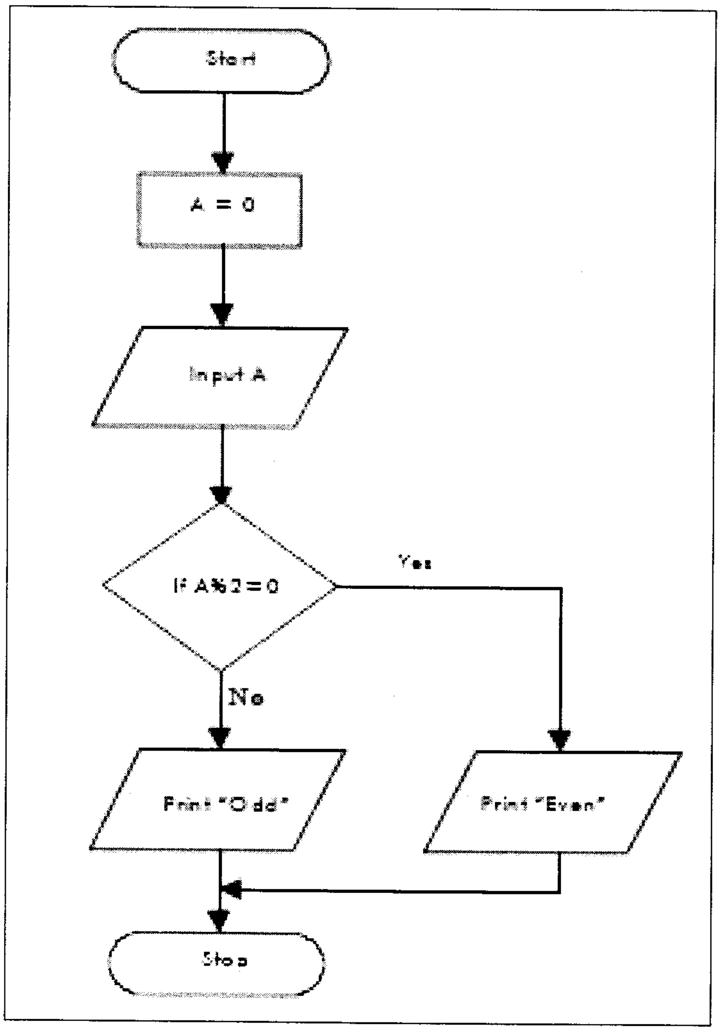


Figure 4.28: Another Example of Programme Flow Chart to check even or odd number

HIPO Diagram

HIPO is another commonly used method for developing systems software. An acronym for Hierarchical Input Process Output, this method was developed by IBM for its large, complex operating systems.

The assumption on which HIPO is based is that, it is easy to lose track of the intended function of a system or component in a large system. This is one reason, why it is difficult to compare existing systems against specifications (and therefore why failures can occur even in systems that are technically well formulated). From the user's view, single functions can often extend across several modules. The concern of the analyst then is understanding, describing and documenting the modules and their interaction in a way that provides sufficient details but that does not lose sight of the larger picture.

HIPO diagrams are graphic, rather than prosaic or narrative, description of the system. They assist the analyst in answering three guiding questions.

- What does the system or module do? (Asked when designing the system.)
- How does it do it? (Asked when reviewing the code for testing or maintenance.)

- What are the inputs and outputs? (Asked when reviewing the code for testing or maintenance.)
- A HIPO description for a system consists of visual table of contents and functional diagrams.

Visual Table of Contents

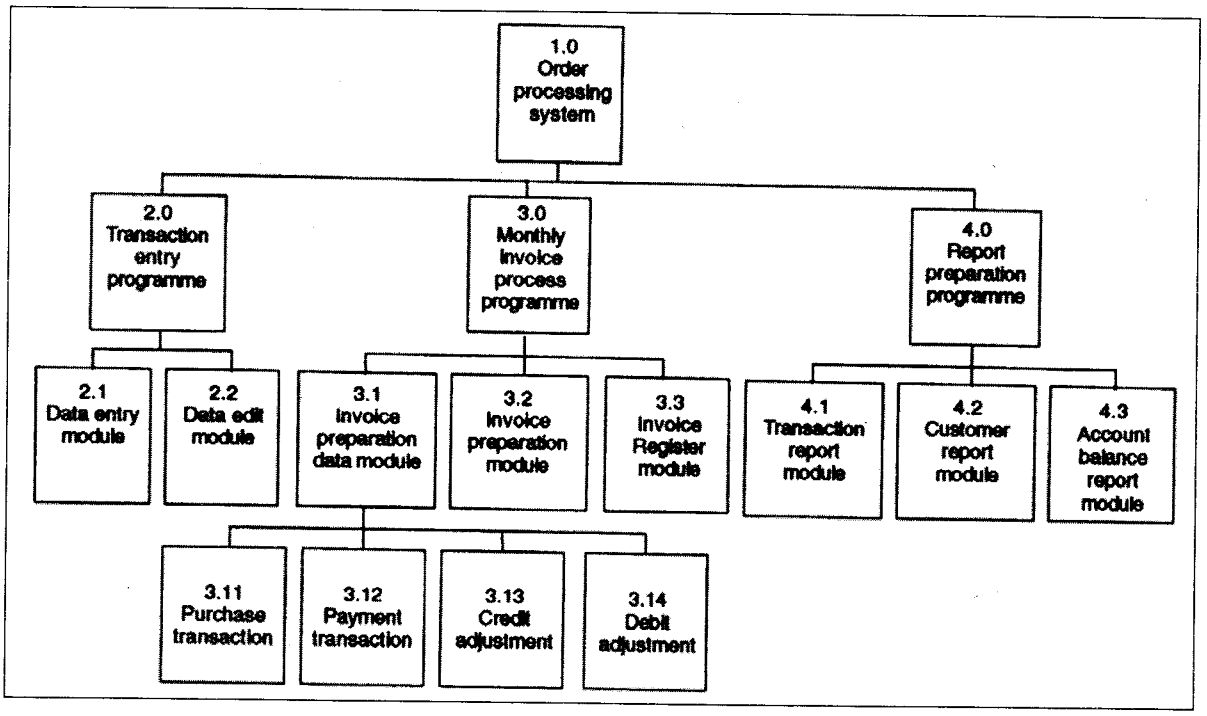
The Visual Table Of Contents (VTOC) shows the relations between each of the documents making up a HIPO package. It consists of a hierarchy chart that identifies the modules in a system by number and in relation to each other and gives a brief description of each module. The numbers in the contents section correspond to those in the organization section.

The modules are in increasing detail. Depending on the complexity of the system, three to five levels of modules are typical.

Functional Diagrams

There is one diagram for each box in the VTOC. Each diagram shows input and output (right to left or top to bottom), major processes, movement of data and control points. Traditional flow chart symbols represent media such as magnetic tape, magnetic disk, and printed output. A solid arrow shows control paths, and an open arrow identifies data flow.

The illustration is an overview of the system, representing the top level box in the VTOC. The numbers on the lower portion of the diagram indicate, where more details will be found; that is, they refer to other document numbers.


1.0 Order processing system
2.0 Transaction entry programme
3.0 Monthly invoice process programme
4.0 Report preparation programme
2.1 Data entry module
2.2 Data edit module
3.1 Invoice preparation data module
3.2 Invoice preparation module
3.3 Invoice Register module
4.1 Transaction report module
4.2 Customer report module
4.3 Account balance report module
3.11 Purchase transaction
3.12 Payment transaction
3.13 Credit adjustment
3.14 Debit adjustment

Some functional diagrams contain other intermediate diagrams (Figure 4.29). But they also show external data, as well as internally developed data, and the steps in the procedure where the data is used. A data dictionary description can be attached to further explain the data elements used in a process.

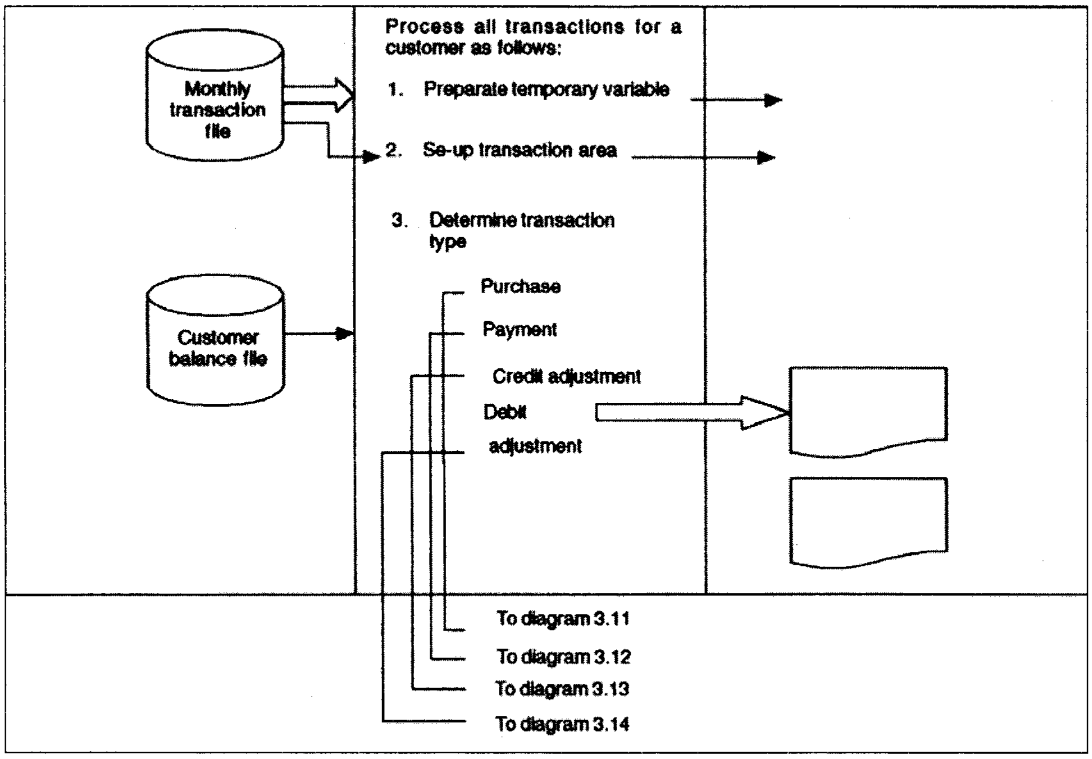


Figure 4.29: HIPO Functional Diagram for Monthly Invoice Processing

HIPO diagrams are effective for documenting a system. They also aid designers and force them to think about how specifications will be met and where activities and components must be linked together. However, they rely on a set of specialized symbols that require explanation and extra concern when compared to the simplicity of, for example, data flow diagram. HIPO diagrams are not as easy to use for communication purposes as many people would like. And, of course, they do not guarantee error free systems. Hence, their greatest strength is the documentation of a system.

Warnier-Orr Diagrams

Wanier-Orr diagrams are other tools aimed at producing, working and correcting program. The Wanier-Orr diagrams take its name from its co-developers, Jean-Dominique Warnier and Kenneth Orr. Unlike VTOCs, pseudocode or flow charts, which read from the top to down and then from left to right, the Wanier-Orr diagram reads from left to right, then from the top to down. Whereas a flow chart requires many symbols, Wanier-Orr diagrams employ brad circles, parentheses, dots, and bars. Diagrams can depict data-dictionary-type definition or detailed program logic.

The Wanier-Orr uses brackets to group related elements, following the sequence control structure. Technically the symbol for a bracket is "[" and a brace is "{"but Wanier-Orr calls "{", a bracket. Thus, we see that three elements in Figure 4.30 make up the single element "systems process". Two elements, preliminary and detailed, make up analysis.

Iteration structure (called repetition in the Warnier-Orr notation) is depicted by a parenthesis to the left of the series of elements to be repeated (Figure 4.31). A number inside the theses indicates the number of times the iteration should be performed. Thus, there will be four bracketed elements, until there are no more bicycles to assemble.

(a) The systems process drawn in Warnier-Orr fashion

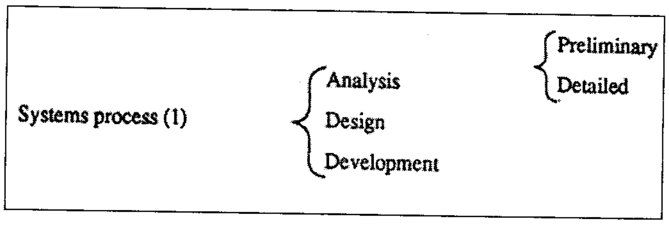


Figure 4.30

(b) Warnier-Orr diagram for bicycle assembly

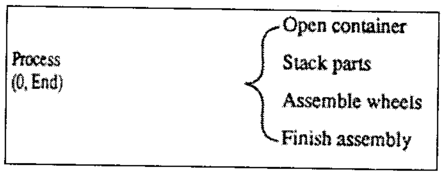


Figure 4.31

(c) Warnier-Orr diagram for the accounts payable stub-over-cheque module. This diagram shows the three control structures of sequence, selection, and repetition

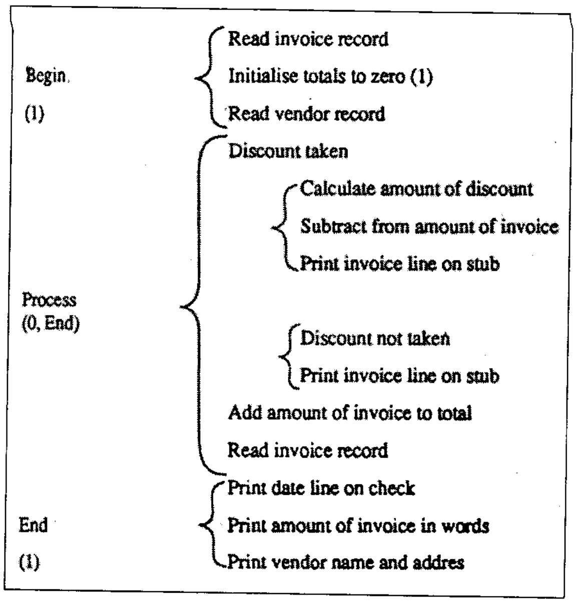

Begin
(1)
Read invoice record
Initialise totals to zero (1)
Read vendor record
Process
(0, End)
Discount taken
Calculate amount of discount
Subtract from amount of invoice
Print invoice line on stub
Discount not taken
Print invoice line on stub
Add amount of invoice to total
Read invoice record
End
(1)
Print date line on check
Print amount of invoice in words
Print vendor name and addres

Figure 4.32

A plus sign enclosed in a circle, indicates an alternation or selection structure. A bar separates the decision into true (above the bar) and false (below the bar) (See Figure 4.32).

For our AP cheque-printing logic, the Warnier-Orr diagram (Figure 4.32) has brackets surrounding the repetitive operations and a decision point inside the process section to determine whether a discount will be taken. This diagram also has a beginning and ending logic that our bicycle example does not require.

Warnier-Orr diagrams show the beginning, processing, and ending parts of the detailed logic, quite explicitly. In keeping with the structured methodology, they have a single entry and a single exit, they support the three control structure and compared with other tools, they employ few symbols.

Disadvantages of the Warnier-Orr system include left-to-right, top to down construction (the opposite of all other tools which are top to down, left to right) and its focus on processing versus data flow.

Introduced by Warnier and later modified by Orr, Warnier-Orr diagrams are thus, used to decompose and partition the contents of a data store, much like DFDs are used to decompose the functions of a system.

Figure 4.33 illustrates how a Warnier-Orr diagram would be drawn for the employee-master file. As before, the employee-master file is defined as a set of employee records, thus, leading to the equation:

However, the Warnier-Orr diagram tells us how to define an employee record. We can write (as before):

Employee_record	=	Employee number+ employee_name_and_ address + personnelinformation + wage. and_salary_information + leave_and_pension _history+current_payroll_history

Let us suppose next, that we, want to know which attributes make up the category employee name and address. The Warnier-Orr diagram informs us that

Employee_name + employee	=	Employee_name+employee_ home_address + employee, plant_address + telephone. extension_number

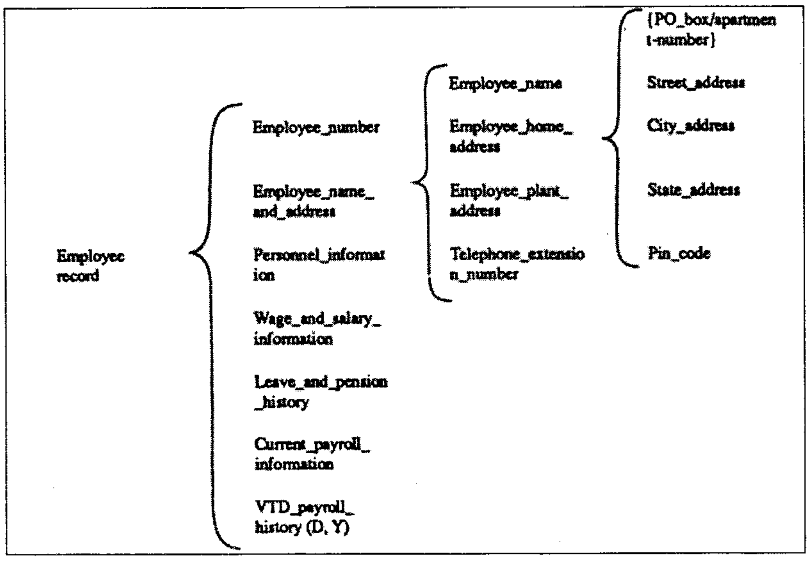

Employee record
Employee_number
Employee_name_ and_address
Personnel_informat ion
Wage_and_salary_ information
Leave_and_pension _history
Current_payroll_ information
YTD_payroll_ history (D, Y)
Employee_name
Employee_home_ address
Employee_plant_ address
Telephone_extensio n_number
{PO_box/apartmen t-number}
Street_address
City_address
State_address
Pin_code

Figure 4.33: Warnier-Orr Diagram Showing Decompostion of the Employee Record

If we wanted to know, which attributes make up employee home address, further decomposition would be necessary. The Warnier-Orr diagram tells us that

Employee_home_address	=	({P0_box_/apartment_number))+ street_address+city_address+ state_address+Pin_code

where the post office box or the apartment number is optional.

A unique feature of Warnier-Orr diagrams is that, they show sequences, either or condition and on repetitions of a process. As illustrated by Figure 3.33, most Warnier-Orr diagrams statements imply a sequential order. For example, employee, name_and_address is equal to the employee_name plus the employee, home_address, plus the employee_plan_,address, plus the telephone_extension_number. Finally, repetition of a process is indicated. Current_payroll_ history is shown as a variable for each employee.

The Warnier-Orr diagrams continue to be pushed to the right until all right-hand attributes can be defined by their field length or physical storage requirements.

City address might be defined as

City_address = 2 (character) 12

or as requiring from two to twelve characters. Telephone extension might be defined as

Telephone_extension = 4 digits,

where an extension always consists of a four digit number.

As this example suggests, there are several good reasons for using Warnier-Orr diagrams to describe the contents of data stores. First, Warnier-Orr diagrams are easy to construct, read, and interpret. Second, they permit larger, complex terms to be decomposed into constituent parts. Third, Warnier-Orr diagrams describe the three logical constructs used in programming. Fourth, they can be used to analyse, both actions and things. Fifth, they simplify the definition and order of terms to be entered into the data dictionary. The main disadvantage of a Warnier_Orr diagram is that it does not show relationships which exist within and between data stores.

Nassi-Shneidermann Charts

Nassi-Shneidermann (N-S) charts offer an alternative to either pseudocode or program flow charts. Named after their authors, N-S charts are much more compact than program flow charts, as it include, pseudocode like statements, feature sequences, decisions, and repetition constructs. Figure 4.34 illustrates an N-S chart designed for processing payroll cheques.

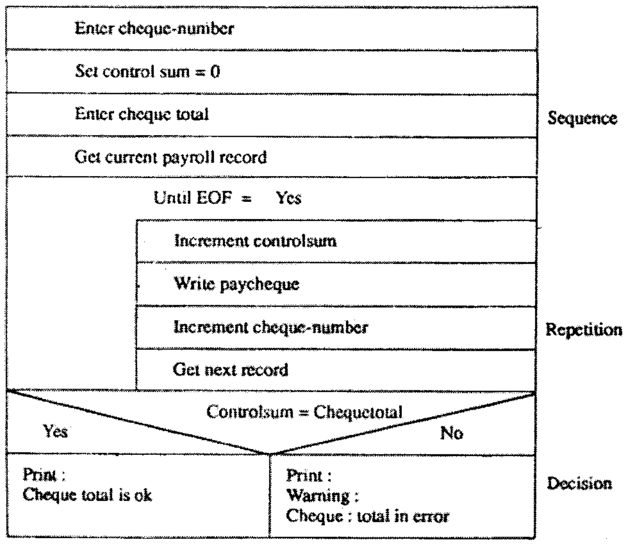


Figure 4.34: Nassi-Shneidermann Chart

Nassi-Sheidermann charts are also sometimes called the Chapin charts. This system was developed by and named for 1. Nassi and B. Shneidermann in the early 1970s and differs markedly from those we have examined so far. It uses rectangles divided into halves with an angular line for selection, a horizontal rectangle for sequence, and the word DOWHILE for iteration. The bicycle-assembly chart appears in Figure 4.35, and the AP check-printing system with discounting in Figure 4.36.

Nassi-Shneidermann charts are winning wider acceptance in the computer industry because they so simply illustrate complex logic. Perhaps, as analysts evaluate the various tools at their disposal, these charts will gain even more followers. As with other structured tools, they are single entry and single exit, and efficiently accommodate modules. However, they are not as useful for conveying system flow as they are for detailing logic development and they are difficult to maintain.

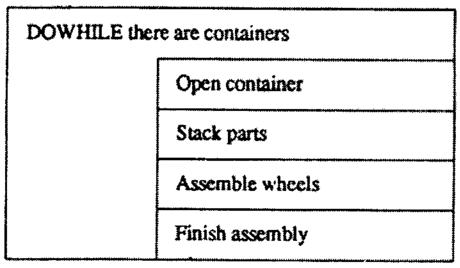


Figure 4.35: Nassi-Shneidermann Chart for Bicycle Assembly

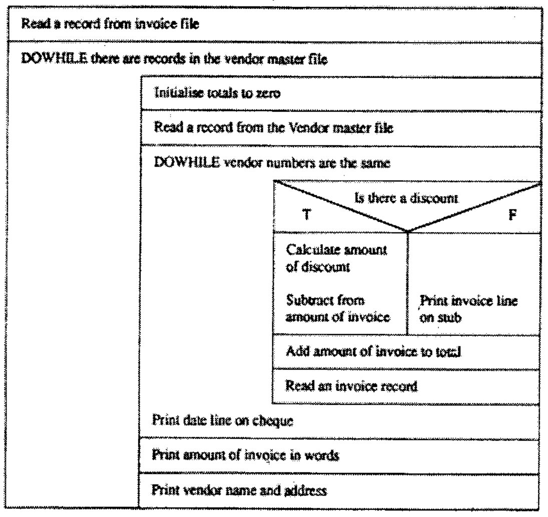

Read a record from invoice file
DOWHILE there are records in the vendor master file
Initialise totals to zero
Read a record from the Vendor master file
DOWHILE vendor numbers are the same
Is there a discount
T
F
Calculate amount of discount
Subtract from amount of invoice
Print invoice line on stub
Add amount of invoice to total
Read an invoice record
Print date line on cheque
Print amount of invoice in words
Print vendor name and address

Figure 4.36: Nassi- Shneidermann Chart for AP Cheque-printing Logic

How to Prepare Documentation

Documentation can be prepared manually or by using computers. Manual methods are always time-consuming and require more manpower. Computers assist analysts in preparing documentation in many ways. During each phase of SDLC, the analysts prepare documents and draw diagrams by using word processor graphics software and Case tools.

All these documents, files and graphics are later set into single file before implementation phase for making each project report. The project reports are designed by using a DTP software and laser printouts are taken. The project reports are then printed, binded and presented in a book form.

Check Your Progress

1. What are process symbols?
2. Discuss Interviews.

4.6 LET US SUM UP

In this lesson we discussed the following:

- Main emphasis in system analysis is on determining the system requirements. System requirements are the various features of the system like inputs, outputs and various processes.
- System requirements are documented and requirements specifications (SRS) are prepared.
- Documentation of requirements is necessary as there is a lot of handing over of requirements from one hand to another.
- The requirements analysis is done through fact finding techniques. These tools of fact finding include interviews, questionnaires, onsite observations and study of procedures and manuals.
- Studying procedures and documents of the system are useful in gathering information.
- Information about various processes may be gained by observing the work of the users.
- Users may be interviewed personally or may respond to a set of questions or a questionnaire. This also helps in gathering of information by an analyst.
- Organization chart is used to represent the hierarchy of the organization. The relations of various departments, divisions, people with each other can be shown using an organization chart.
- Decision tree and table are used to represent the processes where different actions are taken depending on various conditions.
- Decision trees are shown like a branch of a tree with nodes as decision points and the leaves as resulting actions.
- Decision table is made up of four parts - condition statement, condition entry, action statement, action entry. Entries are made in correspondence to a list of conditions and proper action entry marked in correspondence.
- Decision tables are of limited entry (Yes/No), extended entry or mixed entry type.
- Structured English is used to depict sequence, decision condition or iterational structure logics in English like structures.
- System flow chart is used to define a business system as various processes and media are used in it.
- Warnier/Orr diagrams are used in development of complex systems. The key is to move backwards from output to further backward.
- HIPO charts list inputs, outputs and various processes in a graphical format.
- Structure charts are used to represent the relationship between various modules in a software, and also what data they share between each other.

- Processes are defined using DFDs (Data Flow Diagram). The source and destination of data, the data flow, processes and data stores implied are shown.
- Context diagrams are O level DFDs. They have only one process.
- Context diagrams may be further exploded into level 1 DFD's. Each process of level 1 DFD may be further exploded into level 2 DFDs and so on.
- Data dictionary is a repository of all data items used in a system. It may hence be called data about data or metadata.
- Data dictionary contains details of all data elements, the smallest possible units of data, and data structure, set of data elements or other data structures.
- Entities are anything which may be characterized by certain attributes.
- Entities have relations amongst them.
- Entity relationship diagram represents the various entities and the relations.

4.7 KEYWORDS

Decision Trees: Decision trees are shown like a branch of a tree with nodes as decision points and the leaves as resulting actions.

Decision Table: It is made up of four parts - condition statement, condition entry, action statement, action entry.

Structured English: It is used to depict sequence, decision condition or iterational structure logics in English like structures.

System Flow Chart: It is used to define a business system as various processes and media are used in it.

Warnier/Orr Diagrams: These are used in development of complex systems. The key is to move backwards from output to further backward.

HIPO Charts: HIPO charts list inputs, outputs and various processes in a graphical format.

Data Dictionary: It is a repository of all data items used in a system.

Entities: Entities are anything which may be characterized by certain attributes.

Entity Relationship Diagram: It represents the various entities and the relations.

4.8 QUESTIONS FOR DISCUSSION

1. What do you understand by the term "system requirements"? Why do we need to determine system requirements?
2. Compare a structured interview with an unstructured one.
3. What do you mean by onsite observation? Describe by giving an example.
4. Differentiate between the following:
   (a) Interview and questionnaire
   (b) Open-ended and close-ended questions

(c) Rating type and ranking type questions

(d) Physical and logical DFD

(e) Data store and data flow.

(f) Procedures and manuals.

5. What procedure must be used to prepare a questionnaire?
6. What is a data dictionary? Discuss its importance in structured analysis.
7. Name the commonly used process description tools. Discuss the importance of each one in brief.
8. Explain three basic types of structured English statements with suitable examples.
9. What is a decision tree? How does it differ from a discuss table? Explain with examples.
10. What is DFD? Describe the steps of draw, various tools of structured analysis.
11. Describe the various types of fact-finding techniques with examples.

Check Your Progress: Model Answers

1. The process symbols are rectangular boxes that represent a simple process or steps in a program. The process symbol represents initialization of values, input and output operations and calls to execute other procedures.
2. The interview is a face-to-face, interpersonal role situation in which a person called the interviewer asks a person being interviewed, questions designed to get information about a problem area. The interview is the oldest and most often used device for gathering information in systems work.

4.9 SUGGESTED READINGS

Lars Skyttner, *General Systems Theory*, World Scientific.

Ludwig von Bertalanffy, *General System Theory : Foundations, Development, Applications*, New York : George Brazillier, 1969.

LESSON

5

COST BENEFIT ANALYSIS

CONTENTS

5.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Describe the real cost of processing
- Describe the real benefits provided by a system
- Understand cost-benefit analysis
- Identify ad classify cost and benefits
- Define evaluation
- Describe various methods of evaluation
- Define feasibility of the project
- Describe various types of feasibility

5.1 INTRODUCTION

Cost is associated with the two activities of development and operation as follows:

- Development comprises all the stages from the initial investigation stage to the successful handover of a final system to the users.
- Operation also includes maintenance of the system.

It is rare for a computer based system to run for years without substantial changes being needed to allow necessary alternations changing business requirements or improvements in hardware and software. In fact, it is the experience of many long-established computer software firms that the major part of their analysis and programming effort is devoted to the maintenance of existing systems. So, an important objective is to minimize costs in these areas.

Benefits are also of different type and can be grouped on the basis of advantages they provide to the management.

5.2 THE REAL COST

The real cost of processing includes four different categories of possible costs as follows:

(a) ***Hardware Costs:*** The hardware costs include the costs of actual purchase or lease of computers, terminals, storage and output devices, as well as any additional furniture and other equipments.

It is generally more difficult to determine the actual cost of hardware when the system is shared by various users than for a dedicated stand-alone system.

(b) ***Software Costs:*** The software costs are also easy to estimate if we use packaged software for particular applications. There should be quoted price for each application package. The problem becomes more complex if we wish to modify an existing package. This would require quotations from one or more systems design consultants. The most difficult pricing situation occurs when only customised software would match our needs. Nonetheless, we should be able to negotiate costs with our programming consultant. It is recommended that we agree upon a reasonable figure, if possible.

(c) ***Installation and Implementation Cost:*** The hardware and software costs are treated as our initial system cost for tax calculation purposes. In addition, installation and implementation costs, some of which have been itemised below, may be included in the initial pricing. Initial operator training may be included in the system price. However, we may also have to pay for backup personnel in manual processes while the training is in progress.

Some of the heads, which would incur costs, are as follows:

- ❖ Modifications may be required to the room which would house the system.
- ❖ Some systems may require temperature, humidity or dust control equipment to be installed.
- ❖ We may wish to install fire or smoke detectors, fire extinguishers, security locks and the like.
- ❖ Stabilised power supply must be accounted for.
- ❖ Conversion costs of moving from our old system to a new one may arise.
- ❖ Extra programming may be required even with pre-packaged software.

- ❖ Consultant's fees should be included if use their services selecting and implementing our system.

(d) ***Ongoing Support and Growth Costs:*** Even where we cannot estimate a cost accurately, it's wise to include an estimate based on whatever information can be gathered. The support and growth costs are as follow:

- ❖ Maintenance cost, which may be between 10 to 12 percent. Equipment rental is a major expense in case we opt for a pleasing plan.
- ❖ Training of new operators may be required.
- ❖ We may experience idle time cost when our computers are down and operators can't be assigned to other productive tasks.
- ❖ Finance changes on borrowed capital used for system acquisition.
- ❖ Insurance premium to cover the risk of damage to our system and records.
- ❖ Power and utilities charges.
- ❖ Software maintenance contract costs. This service provides continuous scheduled updates and removal of bugs and viruses.
- ❖ Consumable supplies, like paper, printer ribbons, and magnetic storage media (disk or tape), are a significant continuing cost.
- ❖ The cost of upgrading our system to accommodate future growth.

(e) ***Personnel Costs:*** Personnel costs include the salaries and benefits (insurance, vacating time, sick pay, etc.) provided to EDP staff as well as to those involved in developing the system. Costs incurred during the development of a system are one time costs and are known as developmental costs. Once the system has been installed, the costs of operating and maintaining the system become recurring costs.

(f) ***Supply Costs:*** Supply costs are variable costs that increase use of paper, ribbons, disk and the like. They should be estimated and included in the overall cost of the system.

5.3 THE REAL BENEFITS

A system is also expected to provide benefits. The first task is to identify each benefit and then, assign a monetary value to it for the cost-benefit analysis.

The benefits of a project include four types:

(a) ***Cost-Saving Benefits:*** Cost-saving benefits lead to reduction in administrative and operational costs. A reduction in the size of the clerical staff used in the support of an administrative activity is an example of a cost-saving benefit.

(b) ***Cost-avoidance Benefits:*** Cost-avoidance benefits are those which eliminate costs. No need to hire additional staff in future to handle an administrative activity is an example of a cost avoidance benefit.

(c) ***Improved-Service-Level Benefits:*** Improved-service-level benefits are those where the performance of a system is improved by a new computer-based method. Registering a student in fifteen minutes rather than an hour is an example of this type of benefit.

(d) ***Improved-Information Benefits:*** Improved-Information benefits is where computer based methods lead to better information for decision making. For example, a system that reports the most, improved fifty customers, as measured by an increase in sales is an improved-information. This information makes it easier to provide better service to major customers.

5.4 HOW TO DEFINE COST BENEFIT ANALYSIS

We can define cost-benefit analysis as:

(i) That method by which we find and estimate the value of the gross benefits of a new system specification.

(ii) That method by which we find and determine the increased operating costs associated with the above mentioned gross benefits.

(iii) The subtraction of these operating costs from the associated gross benefits to arrive at net benefits.

(iv) That method by which we find and estimate the monetary value of the development costs that produce the above mentioned benefits.

(v) Those methods by which we show the time-phased relationship between net benefits and development costs as they relate to cash flow, payback on investment, and time-in-process taking (or not taking) into operation factors such as inflation etc. In short, the calculation of actual net benefit as cash flowback overtime.

5.4.1 Steps in Cost Benefit Analysis

Cost benefit analysis is the major activity of feasibility study for determining the economic feasibility of the project. It is done in the following steps:

(a) Identification of costs and benefits.

(b) Classifications of costs and benefits.

(c) Selection of evaluation method.

(d) Determining the feasibility of the project.

5.5 IDENTIFICATION OF COSTS AND BENEFITS

First of all, the analyst identifies those costs and benefits of the project, that can be measured. For example, the cost of hardware, system software, stationery, etc. and the savings from reduced costs can easily be identified and measured. The analyst also identifies those costs and benefits which cannot be measured. For example, it is often difficult to measure the cost incurred in providing better customer service and improving company image during implementation of a new system.

5.6 CLASSIFICATION OF COSTS AND BENEFITS

After identifications of various costs and benefits, they are classified. Costs element can be classified into two categories - tangible and intangible costs. Tangible costs are those costs whose values can be precisely determined. For example, equipment costs, material costs (stationery, documentation, etc.), personnel costs (salaries and remunerations), facility costs (air-conditioning, wiring, etc.), operating costs (computer time, etc.) and other costs (consultancy, travelling, etc.) are the major tangible costs. Intangible costs, on the other hand, are those whose values cannot be precisely determined. For example, the cost of breakdown of online system and problems faced by employees during implementation of new system are intangible costs.

Benefits can also be classified as tangible and intangible benefits. Tangible benefits are those savings that can be actually measured. For example, decrease in production costs and increase in sales are tangible benefits. Intangible benefits, on the other hand, cannot be measured. For example, improvement in the company's image due to computerisation is the intangible benefit. Let us review each category:

(i) ***Tangible costs and benefits:*** Tangibility refers to the ease with which, costs or benefits can be measured. For example, an outlay of cash for any specific item or activity is referred to as a tangible cost. The purchase of hardware or software, personnel training and salaries are examples of tangible accounts costs. They are readily identified and measured.

Benefits are often more difficult to specify exactly than costs. For example, suppliers can easily quote the cost of purchasing a terminal but it is difficult for them to tell specific benefits or financial advantages for using it in a system. Tangible benefits such as completing jobs in fewer hours or producing error free reports are quantifiable.

(ii) ***Intangible costs and benefits:*** Costs that are known to exist but whose financial value cannot be accurately measured are corporate known to as intangible costs. The estimate is only an approximation. It is difficult to fix exact intangible costs. For example, employee morale problems because of installing new system in an intangible cost. How much moral of an employee has tree affected cannot be exactly measured in terms of financial values. Intangible costs may be difficult even to identify, such as an improvement in customer satisfaction from a real-time order entry system.

Intangible benefit such as more satisfied customers or an improved corporate image because of using new system are not easily quantified.

Both tangible and intangible costs and benefits should be taken into consideration into evaluation process. If the project is evaluated on a purely intangible basis, benefit exceed costs by a substantial margin, then we will call such project as cost effective. On the other hand, if intangible costs and benefits are included, the total cost (tangible as well as intangible) exceed the benefits which makes the project an undesirable investment. Hence, it is desirable that systems projects should not be evaluated on the basis of intangible benefits alone.

(iii) ***Direct or Indirect costs and benefits:*** Direct costs are those which are directly associate with a system. They are applied directly to the operator. For example, the purchase of floppy for ₹ 400/- is a direct cost because we can associate the floppy box with money spend.

Direct benefits also can be specifically attributable to a given project. For example, a new system that can 30% more transactions per day is a direct benefit.

Indirect benefits are realized as a by-product of another system. For example, a system that tracks sales calls on customers provides an indirect marketing benefit by giving additional information about competition. In this case, competition information becomes an indirect benefit although its work in terms of money cannot be exactly measured.

(iv) ***Fixed or Variable costs and benefits:*** Some costs and benefits remain constant, regardless of how a system is used. Fixed costs are considered as sunk costs. Once encountered, they will not recur. For example, the purchase of an equipment for a computer centre is called as fixed cost as it remains constant whether in equipment is being used extensively or not. Similarly, the insurance, purchase of software etc. In contrast, variable costs are incurred on a regular basis. They are generally proportional to work volume and continue as long as the system is in operation. For example, the cost of computer forms vary in proportion to the amount of processing or the length of the reports desired.

Fixed benefits also remain constant. By using a new system, if 20% of staff members are reduced, we can call it a fixed benefit. The benefit of personnel saving may occur every month. Variable benefits, on the other hand, are realized on a regular basis. For example, the library information system that saves two minutes in providing information about a particular book whether it is issued or not, to the borrower compared with the manual system. The amount of time saved varies with the information given to the number of borrowers.

5.7 EVALUATION

Computerised information systems are developed and utilized by two categories of organisations:

(a) firms, which have the in-house capability.

(b) Service bureaus, which develop them for usage by the outside clients.

In both cases, the basic investments are of a high order in terms of not only the computer system, but also in terms of site preparation involving air-conditioning, civil and electrical works followed by recruitment of manpower (computer-centre manager, system analysts, programmers and operators beside input/output, quality control, data preparation and other support staff) and their training. There can be only one objective behind making such sizeable investments and that is to provide satisfaction to the end user, in-house or outside.

(i) *System Cost:* In most cost-conscious organisation, an initial estimate is prepared for the on-line cost of developing the system and the recurring costs of running the system. The cost estimation has to cover details such as:

- Routine manpower (systems, programming, operations and for service bureau, marketing staff).
- Manual manpower (specially employed to handle production, quality control, correction of check-bots etc.)
- Data preparation (direct entry or punch entry or punch lord kind, whether done in house or by outside agencies)
- Consumable stores (Stationery, cards or floppy disk, carbon, ribbons etc.)
- Computer time (actual usage house often logged by the computers itself)
- Administrative expenses
- Logistic expenses (for the conveyance of manpower, transportation of documents or output report etc.)
- Miscellaneous expenses (Overtime etc.)

(ii) ***Need for Evaluation:*** Whether or not service bureaus face their irritated customers or user organisations meet their dissatisfied departmental heads, a sound principal to run computerised information systems is to introduce a reliable procedure for management of hardware, software and data preparation. Such practices, as regular and time bound squares and components, do help hardware management. Attention to programming capability, scientific design and development of systems and a high-quality support for system software is invaluable for software management. Data preparation is a week link in most of the organisations and quality control of entered data is a music, among other factors.

An actual evaluation plan has to begin from the stage when one knows what has been spent on a computerised information system and then, projected to find out whether there has been value for money spent or not. Such an evaluation is certainly not easy and can be approached in a twofold manner the realization from the process side of system evaluation and from the product side of output reports.

(iii) ***Process Evaluation:*** A process evaluation is carried out from the computer professional's point of view. Design of the system and the quality of programming have to stand the rigours of careful assessment. Quite often the system design in presented by the project leader to the entire application software group and gains from their friendly criticism. Programming standards are today quite high and a modular approach is far preferable to single integrated programmes. Internal or internal training in efficient program writing techniques can achieve surprisingly good results.

Another aspect of process evaluation is the utilizations of hardware resources. In all computers capable of rumming multiple programs there should be adequate prior consideration to arrive at different memory partitions, and to allocate input-output devices in a judicious way to each partition. The allocation of certain tapes or discs to production or development jobs often helps in obtaining an efficient and steady mix of jobs. The test for evaluation is to ensure optimal system utilizations, with the least possible idleness of any single device.

The third aspect of process evaluation is to check whether there is minimum wastage of computer-time. It may happen that well-designed systems with good quality programmes are running with an apparently maximum waste of hardware resources and still they may hide many wasteful runs. This arises due to two reasons associated with development and production stages of the information system. At the development stage, lack of rigorous quality control may allow many avoidable runs of the programmes. At the production stage lack of full scale debugging may make some programs prone to repeated runs. In fact the best relevant check is to lay down permissible number of development runs and ensure fitness of the programmes for release for production runs without wasting systems resources.

(iv) ***Product Evaluation:*** The product evaluation is concerned with the end user and has to ensure that the output reports are of acceptable quality are continue to be of use. Instances are not that computer outputs, which have long from out of use, are not pointed out as such by managers(users) out of difference to the higher level policy of computerisation while the managers continue to use their little pocket-books containing relevant data. To avoid such a possibility, organisations having a fairly long tradition of computer based information systems should, once in a while take stock of the existing axutomation and computerised operations.

The objective was to devise a questionnaire for each uses department, outlining the group of computer applications for these. General questions related to the usefulness, quality level and achievement of promised improvement and responses were asked on a three tier basis. Specific questions were also framed regarding the reported items of information, frequency of reports, nature of formats and reporting levels. The purpose of the questionnaire was to elicit frank responses from the managers about the utility of the prevailing computerised information systems.

The replies received were then tabulated and put up to the higher management for evaluation of each computerised system from three angles, should the system be continued as such? Or should the system be curtailed or even replaced altogether by other some useful systems? Or should the system be modified to cover more ground so that its utility were enhanced?

The result of such an introspection are not always as per expectations and managers do not feel comfortable to answer such questions or the questions themselves are not formulated clearly or followed up seriously. These considerations however do not belittle their usefulness.

5.8 SELECTION OF EVALUATION METHOD

The common methods of evaluating the costs and benefits are:

(i) Payback Method

(ii) Present Value Method

(iii) Net Benefit Method

(iv) Break-even Method

We will now discuss all these methods in detail alongwith suitable examples.

(i) ***Payback Method:*** The payback method of evaluating the costs and benefits is a common method to determine the time when the accumulated benefits will equal the initial investment. With this method, the analyst knows the time, when the money spent on the project will be recovered. The payback period is, generally, calculated by using the following formula:

$$\text{Payback Time} = \frac{\text{Overall Cost Quality + Installation Period}}{\text{Annual Cash Return}}$$

$$= \frac{(A \times B) + (C \times D) + G}{E + F}$$

Where A is Capital Investment

B is Investment Credit Difference (%)

C is Cost Investment

D is Company's federal income tax bracket

E is Benefits after federal income tax

F is Depreciation

G is Installation Period

An Example

In developing 'Stock Monitoring System' (as discussed in our case scenario), the vice president of the company requested a cost/benefit analysis. The systems analyst identified the various costs and benefits of the project and computed the following data:

(i)	Capital Invested on purchase of hardware and software	₹ 1,00,000
(ii)	Benefits	₹ 2,00,000
(iii)	Investment Credit	10%
(iv)	Cost Investment	₹ 20,000
(v)	Company's Income Tax Bracket	40%
(vi)	Local Taxes	4%
(vii)	Installation period	1 Year
(viii)	Expected life of capital	5 Years

Calculation of the payback period:

Capital Investment (A)	= ₹ 1,00,000
Investment Credit Difference (B)	= 100% - 10% = 90%
Cost Investment (C)	= ₹ 20,000
Company's Federal Income Tax Bracket (D)	= 100% - 40% = 60%

Benefits after federal Income tax (E) are calculated as follows:

Benefits before federal income tax

= Benefits - (Depreciation + Local Taxes)

= 2,00,000 - ((Capital/Life) + Local Taxes)

= 2,00,000 - ((2,00,000/5) + 2,00,000*0.04)

= 2,00,000 - (40,000 + 8,000)

= 2,00,000 - 48,000 = 1,52,000

Benefits after federal income tax

= Benefits before federal income tax - (Benefits before federal income tax XD)

= 1,52,000 - (1,52,000 × 0.60)

= 60,800

Depreciation(E) = 2,00,000/5 = ₹ 40,000

$$\text{Payback Time} = \frac{(1,00,000 \times 0.90) + (20,000 \times 0.60)}{60,800 + 40,000} + 1$$

$$= \frac{1,02,000}{1,00,800} + 1 = 1.01 + 1 = 2.01 \text{ years}$$

Thus, after 2 years, the money spent on the project will be recovered. Hence, the project is economically feasible.

(ii) ***Present Value Method:*** The payback method has certain drawbacks. The values of today's money and tomorrow's money are not the same. In payback method, today's cost is compared with tomorrow's benefits and, thus, the time value of money is not considered. The present value method compares the present values to future values by considering the time value of invested money. The present value is computed with following formula:

$$P = \frac{F}{(1 + r/100)^n}$$

where P is the present value;

F is the future value;

r is the rate of interest; and

n is the number of years.

The project is considered to be economically feasible if the project cost is less than or equal to the present value.

Now, let us understand this method by the following example.

An Example

Suppose, the average annual benefit is ₹ 20,000 for a project of life 5 years. The money is invested by considering the interest rate to be 10 per cent. By using the above formula, the present value for each year is calculated as shown in Table 5.1. The calculation of present value after 2 years is illustrated below:

Given, F = ₹ 20,000

r = 10

n = 2

Then,

$$P = \frac{20,000}{(1+10/100)^2} = ₹\ 18181.82$$

The Break-even is the time when costs of current and candidate systems become equal.

We can conclude that the project would be economically feasible for getting the benefit of ₹ 20,000 after 1 year if the project cost is less than or equal to ₹ 18,181.82 (present value).

Table 5.1: Calculations of Present Value for a Software Project of Life 5 Years

Year	Estimated Future Value Per Year	Present Value of Benefits
1	20,000	18,181.82
2	20,000	16, 528.92
3	20,000	15,026.30
4	20,000	13,660.27
5	20,000	12,410.42

(iii) ***Net Benefit Method:*** The net benefit method is the simplest method of cost/benefit analysis. In this method, the net benefit is calculated by subtracting the total estimated cost from the total estimated benefit. Although it is the easiest method, its has main drawback is that it does not consider the time value of money.

Table 5.2: Calculations of Net Benefits by Subtracting Estimated Costs from Estimated Net Benefits

Year	Estimated Benefit	Estimated Costs	Estimated Net Benefits
1	0	10,000	- 10,000
2	2,000	10,000	- 8,000
3	9,000	10,000	- 1,000
4	20,000	10,000	10,000
5	25,000	10,000	15,000
Total	56,000	50,000	6,000

(iv) ***Break-even Method:*** Break-even method is another useful method of cost/benefit analysis, that is based on the principle of payback method. As in payback method, the costs and accumulated benefits are compared to find the time when the money invested is recovered, similarly, in break-even method, the costs of current and candidate system are compared to find the time when both are equal. This point is called the break-even point. The period prior to break-even is the investment period and the period beyond break-even is the return period.

Table 5.3: Illustration of Break-even Point based on Cost Estimates of Current and Candidate Systems

Year	Cost of Current Systems	Cost of Candidate System	Period Time
1	5,000	10,000	Investment
2	10,000	12,500	Investment
3	15,000	15,000	Break-even
4	20,000	17,500	Return
5	25,000	20,000	Return

5.9 FEASIBILITY OF THE PROJECT

Feasibility is the determination of whether a project is worth doing. The process followed in making this determination is called a feasibility study. This type of study determines if a project can and should be taken. Once it has been determined that a project is feasible. The analysist can go ahead and prepare at the project specification which finalises project requirements. Generally, feasibility studies are undertaken within tight time constraints and normally culminate in a written and oral feasibility report. The contents and recommendations of such a study will be used as a sound bases for deciding whether to proceed, postpone or cancel the project. Thus since the feasibility study may lead to the commitment of large resources, it becomes necessary that it should be conducted competently and that no fundamental errors of judgement are made.

5.9.1 Types of Feasibility

In the conduct of the feasibility study, the analyst will usually consider seven distinct, but inter-related types of feasibility. They are:

1. ***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 location.

   Out of all types of feasibility, technical feasibility generally is the most difficult to determine.

2. ***Operation Feasibility:*** It is mainly related to human organisational and political aspects. The points to be considered are:

   ❖ What changes will be brought with the system?

   ❖ What organisational structures 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?

Generally project will not be rejected simply because of operational infeasibility but such considerations are likely to critically affect the nature and scope of the eventual recommendations. This feasibility study is carried out by a small group of people who are familiar with information system techniques, who understand the parts of the business that are relevant to the project and are skilled in system analysis and design process.

3. ***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 out weigh 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 has a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system life cycle.

   A number of approaches for assessing the costs of solutions have been suggested. Approaches include the following:

   - ***Last cost:*** This is based on the observation that costs are easier to control and identify the revenues. Thus, it assumes that there is no change in income caused by the implementation of a new system. In such an evaluation, only the costs are listed and the option with the lowest cost is selected.
   - ***Time to Payback:*** This method of economic evaluation is an attempt to answer the question. How long would it be until we get out money back on this investment in system? This requires data on both costs and benefits. This method of evaluation has two significant disadvantages:
     (i) It only considers the time taken to return the original investment and ignores the system's long term profitability.
     (ii) The method does not recognize the time value of money. Benefits that accrue in the distant future are not worth as much as similar benefits that occur more quickly but this method fails to recognize this.
   - ***Cost-effectiveness:*** Some type of cost benefit analysis is performed for each alternative. Rough projections of equipment requirements and costs, operational costs, manpower costs, maintenance cost, etc., need to be made. Projections of potential, tangible as well intangible benefits are also needed to be made. For example, tangible benefits are: ability to obtain information, which was previously not available, faster or timely receipt of information, improved or better decision making, improvement in planning and control etc.

4. ***Social Feasibility:*** Social feasibility is a determination of whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change.

5. ***Management Feasibility:*** It is a determination of whether a proposed project will be acceptable to management. If management does not accept a project or gives a negligible support to it, the analyst will tend to view the project as a non-feasible one.

6. ***Legal Feasibility:*** Legal feasibility is a determination of whether a proposed project infringes on known Acts, statutes, as well as any pending legislation. Although in some instances the project might appear sound, on closer investigation it may be found to infringe on several legal areas.

7. ***Time Feasibility:*** Time feasibility is a determination of whether a proposed project can be implemented fully within a stipulated time frame. If a project takes too much time it is likely to be rejected.

Check Your Progress

1. Fill in the blanks:
   (a) .... costs are those costs whose values can be precisely determined.
   (b) With .... method, the analyst knows the time, when the money spent on the project will be recovered.
   (c) Break-even method is based on the principle of .... method.
   (d) Decrease in production cost is an example of .... benefit.
   (e) The .... method compares the today's cost with tomorrow's benefits.
2. State true or false:
   (a) It is difficult to measure the cost incurred in providing better customer service.
   (b) Improvement in the company's image due to computerisation is an intangible benefit.
   (c) The present value method does not consider the time value of invested money.
   (d) The net benefit method considers the time value of money.
   (e) The period prior to break-even is the investment period.

5.10 LET US SUM UP

Cost is associated with development and maintenance. An important objective is to minimize costs in these areas. The real cost of processing includes Hardware costs, Software costs, Installation and Implementation cost, Ongoing support and growth costs, Personnel Costs and Supply cost.

A system is also expected to provide benefits. The benefits of a project include cost-saving benefits, cost-avoidance Benefits, Improved-Service-level benefits and Improved-Information benefits. Cost/benefit analysis is the major activity of feasibility study for determining the economic feasibility of the project. It is done by Identification and classification of costs and benefits, selection of evaluation method and determining the feasibility of the project.

The common method of evaluating the costs and benefits are Payback Method, Present Value Method, Net benefit Method and Break-even Method.

Feasibility is the determination of whether a project is worth doing. The process followed in making this determination is called a feasibility study. Feasibility studies are undertaken within light time constraints and normally culminate in a written and oral feasibility report. Seven distinct, but inter-related types of feasibility. They includes Technical feasibility, operation feasibility, Economic feasibility, Social feasibility, Management feasibility, Legal feasibility, Time feasibility.

5.11 KEYWORDS

Hardware Costs: The costs of actual purchase or lease of computers, terminals, storage and output devices, as well as any additional furniture and other equipments.

Cost-saving Benefits: Benefits that leads to reduction in administrative operational costs. A reduction in the size of the clerical staff used in the support of an administrative activity is an example of a cost-saving benefit.

Cost-avoidance Benefits: Benefit which eliminate cost is known as cost-avoidance benefits.

Improved-Service-Level Benefits: Benefits where the performance of a system is improved by a new computer based method.

Improved-Information Benefits: Benefits where the computer based method lead to better information for decision making.

Tangible Benefits: Those savings that can be actually measured.

Intangible Benefits: Those benefits that cannot be measured.

Intangible Costs: Costs that are known to exist but whose financial value cannot be accurately measured are corporate.

Direct Cost: Those costs which are directly associate with a system.

Fixed costs and Benefits: Costs and benefits that remain constant, regardless of how a system is used.

Variable Benefits: Benefits that are realized on a regular basis.

Feasibility: The determination of whether a project is worth doing.

Technical Feasibility: Specifying equipment and software that will successfully satisfy the user requirement.

Economic Feasibility: To determine the benefits and savings that are expected from a proposed system and compare them with costs.

Social Feasibility: A determination of whether a proposed project will be acceptable to the people or not.

Management Feasibility: A determination of whether a proposed project will be acceptable to management.

Legal Feasibility: A determination of whether a proposed project infringes on known Acts, statutes, as well as any pending legislation.

Time Feasibility: A determination of whether a proposed project can be implemented fully within a stipulated time frame.

5.12 QUESTIONS FOR DISCUSSION

1. Name different types of costs and benefits.
2. Give a good definition of cost-benefit analysis.
3. What is cost/benefit analysis? Enumerate the major steps of this analysis.
4. What is payback method of cost evaluation? Determine the feasibility of a project based on this method for following data:

Capital Invested on Computer	₹ 2,00,000
Benefits	₹ 3,00,000
Investment Credit	10%

Cost Investment	₹ 50,000
Company's Income Tax Bracket	40%
Local Taxes	10%
Installation Period 2 years	
Expected life of capital	4 years

5. Consider the average annual benefit is ₹ 50,000 for a project of life 4 years. The money is invested by considering the interest rate to be 10 per cent. Calculate the present value of money for each year.
6. What is the need for evaluation?
7. List common methods of evaluating costs and benefits.
8. Write short notes on the following:
   (i) Payback Method
   (ii) Present value method
   (iii) Net benefit method
   (iv) Break-even method
9. What do you mean by feasibility of the project?
10. Describe different types of feasibility.

Check Your Progress: Model Answers

1. (a) Software
   (b) Payback
   (c) Payback
   (d) Tangible
   (e) Present Value
2. (a) True
   (b) True
   (c) False
   (d) False
   (e) True

5.13 SUGGESTED READINGS

William N. Sweet, Alexander Kossiakoff, *Systems Engineering*, Wiley-IEEE.

Patrick McDermott, *Zen and the Art of Systems Analysis*, iUniverse.

Robert Sugden, Allan M. Williams, *The Principles of Practical Cost-Benefit Analysis*, Oxford University Press.

UNIT III

LESSON

6

SYSTEM DESIGN

CONTENTS

Contd....

6.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Discuss logical and physical design
- Discuss structured design
- Discuss input and output design
- Discuss form design
- Discuss file structure

6.1 INTRODUCTION

System design is the pivotal point in the system development life cycle. Prior to this phase user requirements have been identified. Information has been gathered to verify the problem and evaluate the existing system. The design is a solution, a "how to" approach, compared to analysis which is a "what is" orientation.

The objectives of Information System Design can be outlined as follows:

- ***Specify the logical design elements***

  System design involves description of detailed design that describes the features of an information system: input, output, files, and databases and procedures.

- ***Support business activities***

  A fundamental objective in the design of an information system is to ensure that it supports the business activity for which it is developed. In other words, the computer and communication technology specified in the design should always be secondary to the results the system is intended to produce.

For example, if it is essential for an organization to move information very quickly to remain competitive, then the design specification of the information system must be built around this essential business objective.

Similarly, the design must fit the way the firm does business. If a sales system is designed to work best for orders that are paid in cash when in fact the firm offers a liberal "sales-on-credit" policy, management will not be happy. These examples show that fitting of a system to the needs of the business is important. Objectives should guide virtually all system design decisions.

- ***Ensure that system features meet user requirements***

  User requirements are translated into system characteristics during design. We say that an information system meets user needs if it accomplishes the following:

  - Performs the right procedures properly.
  - Presents information and instructions in an acceptable and effective fashion.
  - Produces accurate results.
  - Provides an acceptable interface and method of interaction.
  - Is perceived by users as a reliable system.

- ***Easy to use***

  Do the benefits of not using the information system exceed those achieved by relying on the system? This may appear to be an illogical question since the correct information is always preferred to the incorrect. But many technical features of system reliability, accuracy and processing speed are secondary to the human aspects of a system design. Therefore, every analyst strives to design the system to be engineered for people and to include economic features.

- ***Provide software specifications***

  Like the features of an information system, software must also be carefully designed. System design includes formulating software specifications.

  The specifications state input, output and processing functions and algorithms used to perform them. Software modules and routines focusing on what functions each performs and the procedures for accomplishing them are specified as well. Selection of programming languages, software packages and software utilities occur during the logical process and recommendations are included in software specifications.

- ***Confirm to design standards***

  As you can see, the objectives of system design are broad and affect many aspects of both the application and the organization in which the system will be used. As a result, it should be no surprise to learn that well managed information system groups also maintain systems development standards. System design specifications are developed within these standards.

6.2 CONSTRAINTS

The designer, normally, will work under following constraints:

Hardware	: The existing hardware will obviously affect the system design.
Software	: The available software (operating system, utilities, language etc.) in the market will constrain the design.
Budget	: The budget allocated for the project will affect the scope and depth of design.
Time-scale	: The new system may be required by a particular time (e.g., the start of a financial year). This may put a constraint on the designer to find the best design.
Interface	: The new system may require some data from another computerized system or may provide data to another system, in which case, the files must be compatible in format and the system must operate with a certain processing cycle.

6.3 LOGICAL AND PHYSICAL DESIGN

6.3.1 Logical Design

System design goes through two phases of development: logical and physical design. As we saw in the previous chapter, a data flow diagram shows the logical flow of a system and defines the boundaries of the system. For a candidate system, it describes the inputs (source), outputs (destination), data bases (data store) and procedures (data flows) - all in a format that meets the users requirements. When analysts prepare the logical system design, they specify the user needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources. The design covers the following:

- Reviews the current physical system, it's data flows, file content, volumes, frequencies, etc.
- Prepares input specifications. That is, it determines the format, content and frequency of reports, including terminal specifications and locations.
- Prepare edit, security and control specification. This includes specifying the rules for edit correction, backup procedures and the controls that ensure processing and file integrity.
- Specifies the implementation plans.
- Prepares a logical design walk through the information flow, output, input, controls, and implementation plan.
- Reviews benefits, costs, target dates and system constraints.

As an illustration, when safe deposit tracking system is designed, system specifications include weekly reports, a definition of boxes rented and boxes vacant and the summary of the activities of the week - boxes closed, boxes drilled and so on. The logical design also specifies output, input, file and screen layout. In contrast, procedure specifications show, how data are entered, how files are accessed and how reports are produced. (See Figure 6.1).

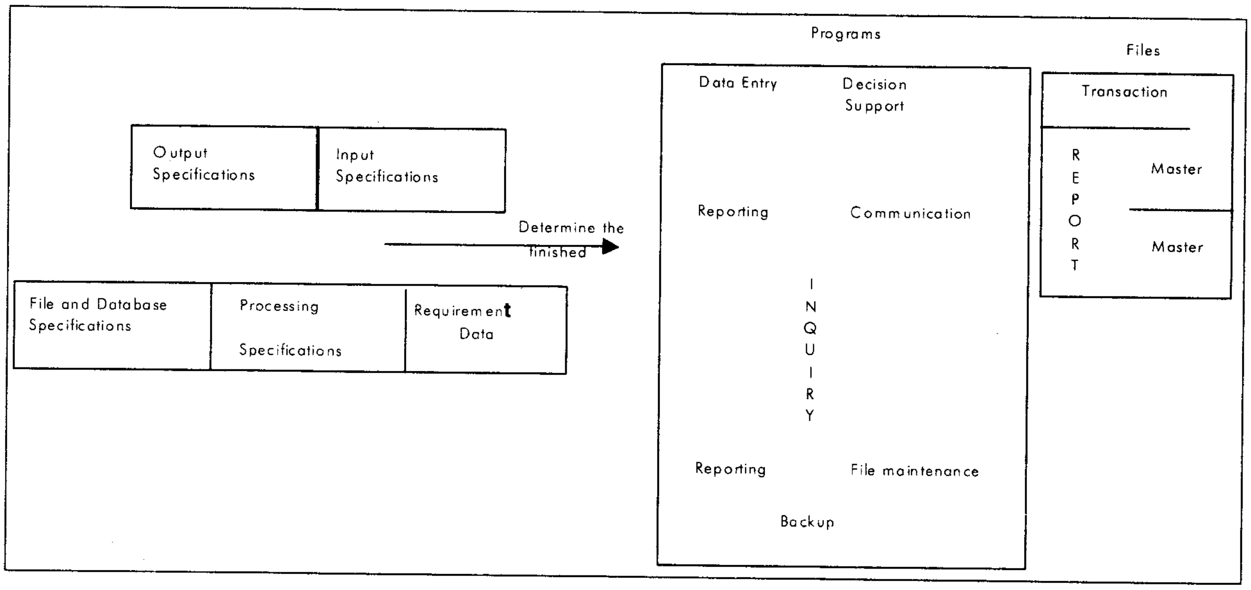


Figure 6.1: Design of System Consists of Logical and Physical Design

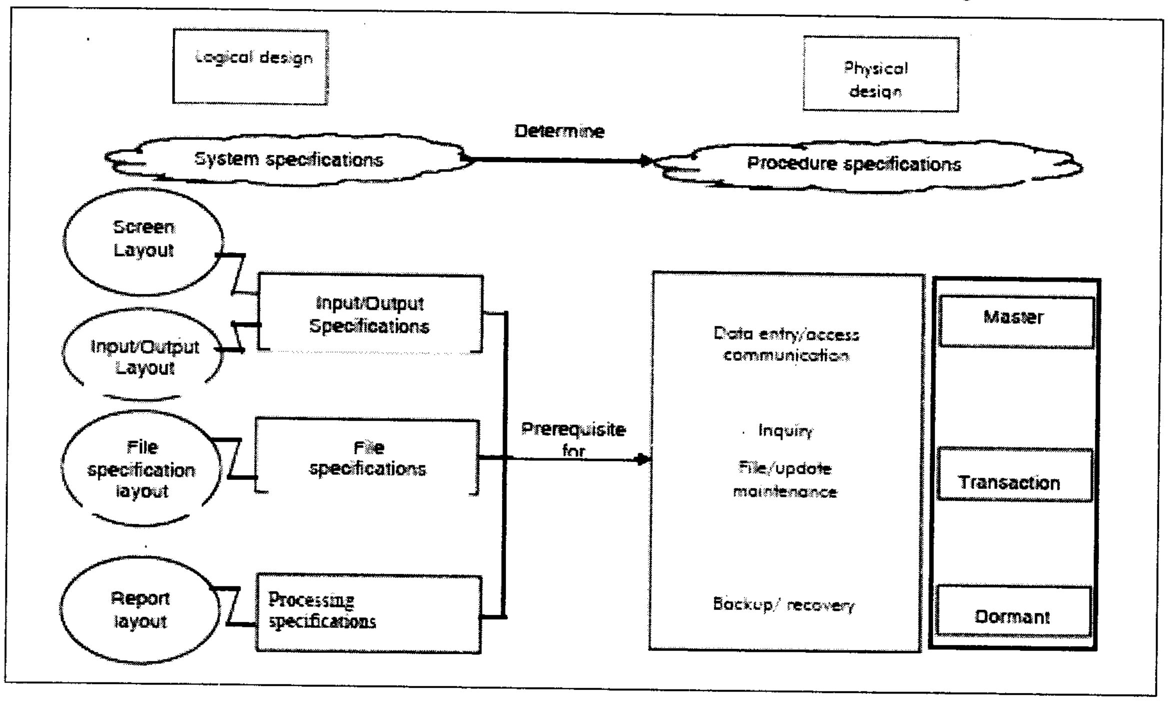


Figure 6.2: Another Representation of Logical and Physical Design

6.3.2 Physical Design

Following the logical design is physical design. This produces the working system by defining the design specifications that tell programmers exactly what the candidate system must do. In turn, programmers write the necessary programs or modifies the software package that accepts input from the user, performs the necessary calculations through the existing file database, produce the report on a hard copy or displays it on a screen and maintains an updated database of all times. Specifically, a physical system design consists of the following steps:

- Design the physical system
  - ❖ Specify input/output media
  - ❖ Design the database and specify backup procedures
  - ❖ Design physical information flow through the system and a physical design walkthrough.
- Plan system implementation
  - ❖ Prepare a conversion schedule and a target date
  - ❖ Determine training procedure, courses and time table
- Devise a test and implementation plan and specify any new hardware/software
- Update benefits, costs, conversion date and system constraints (legal, financial, hardware, etc.)

The physical design for our safe deposit illustration is a software package written in Pascal (a programming language). It consists of program steps that accept new box rental information, change the number of boxes available with every new box rental, print a report by box type, box size and box location and stores the information in the database for reference. The analyst instructs the software programmer to have the package display a menu that specifies for the user how to enter a new box rental, produce a report, or display various information on the screen. These and other procedure specifications are tested and implemented as a working model of the candidate system.

There are three main types of designing, i.e., modular designing, bottom up technique and structured designing. Out of the three, structured designing technique is the best.

6.4 STRUCTURED DESIGN

Structured design is a data-flow based methodology. The approach begins with a system specification that identifies inputs and outputs and describes the functional aspects of the system. The specifications, then, are used as a basis for the graphic representation. The next step is the definition of the modules and their relationships to one another in a form called a structure chart, using a data dictionary and other structured tools.

Logical design proceeds from the top-down. General features, such as, reports and inputs are identified first. Then each is studied individually and in more detail. Hence, the structured design partitions a program into small, independent modules. They are arranged in a hierarchy that approximates a model of the business area and is organized in a top-down manner. Thus, structured design is an attempt to minimize the complexity and make a problem manageable by subdividing it into smaller segments, which is called modularization or decomposition. In this way, structuring minimizes intuitive reasoning and promotes maintainable provocable systems.

A design is said to be top-down if it consists of a hierarchy of modules, with each module having a single entry and a single exit subroutine.

6.5 MODULARIZATION

In structure design a program is segmented into small, independent modules. These are arranged in a hierarchy that approximates a model of the business area and is organized in a top-down manner with the details shown at the bottom. Thus, in structured design, we try to minimize the complexity of the problem and make it manageable by sub dividing it into smaller segments, which is called modularization or decomposition. This has been shown in as below:

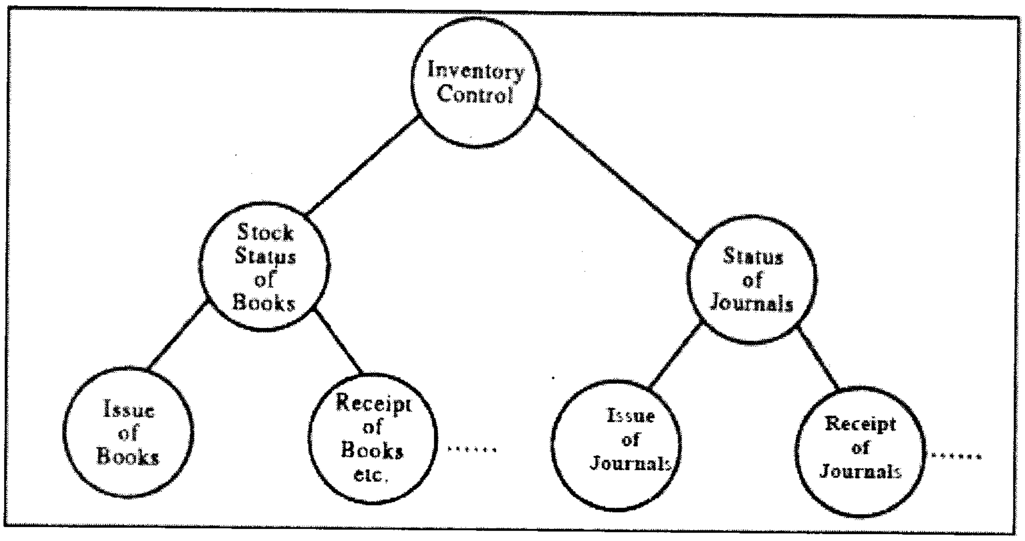


Figure 6.3: Modularization - A Framework

6.5.1 Modular Design

In modular design technique, the software is divided into separately named and addressable components, called modules, that are integrated to satisfy problem requirements. It has been stated that "modularity is the single attribute that allows a program to be intellectually manageable". Monolithic software (i.e., a large program comprised of a single module) cannot be easily grasped by a reader. The number of control paths, span of reference, number of variables and overall complexity would make understanding close to impossible.

Modularization is based on a "divide and conquer" policy - it's easier to solve a complex problem when you break it into manageable pieces. As Figure 6.4 shows, the effort (cost) to develop an individual software module does decreases, as the total number of modules increases, given the same set of requirements - more modules means smaller individual size. However, as modules grow, the effort (cost) associated with integrating the modules also grows. These characteristics lead to a total cost or effort curve in the shown figure. There is a number, M, of modules that would result in minimum development cost but we do not have the necessary sophistication to predict M with assurance.

The curves shown in Figure 6.4 do provide useful guidance when modularity is considered. We should modularize but care should be taken to stay in the vicinity of M. Undermodularity or overmodularity must be avoided.

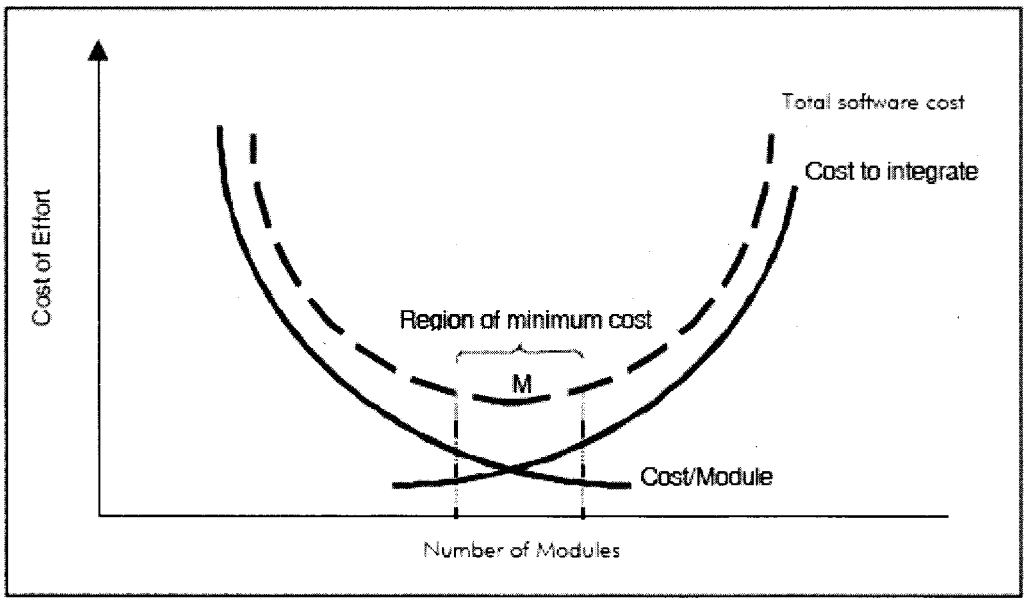


Figure 6.4: Modularity and Project Cost

Another important question arises when modularity is considered – how do we define an appropriate module of a given size? The answer lies in the method(s) used to define modules within a system. Five criteria that enable us to evaluate a design method with respect to its ability to define an effective modular system are discussed below:

- ***Modular Decomposability:*** If a design method provides a systematic mechanism for decomposing the problem into sub-problems, it will reduce the complexity of the overall problem, thereby, achieving an effective modular solution.
- ***Modular Composability:*** If a design method enables existing (reusable) design components to be assembled into a new system, it will field a modular solution that does not reinvent the wheel.
- ***Modular Understandability:*** If a module can be understood as a stand alone unit (without reference to other modules) it will be easier to build and easier to change.
- ***Modular Continuity:*** If small changes to the system requirements result in changes to individual modules, rather than system-wise changes, the impact of change-induced side effects will be minimized.
- ***Modular Protection:*** If an aberrant condition occurs within a module and its effects are constrained within that module, the impact of error-induced side effects will be minimized.

Finally, it is important to note that, a system may be designed modularly even if it's implementation must be "monolithic". There are situations (e.g., real time software) in which relatively minimal speed and memory overhead introduced by sub-programs is unacceptable. In such situations, software can and should be designed with modularity as an overriding philosophy.

6.6 PRINCIPLES THAT GUIDE SOFTWARE DESIGN

The following principles should guide software design:

6.6.1 Modularity and Partitioning

As we discussed in the above section, each system should consist of a hierarchy of modules. Lower level modules are generally smaller in scope and size compared to higher level modules and serve to partition processes into separate functions. Higher modules must "call" lower level modules. For this, some data must be passed on to lower level modules. After performing its function, the relationship is represented by a structure chart. Figure 6.5 is an example of structure chart.

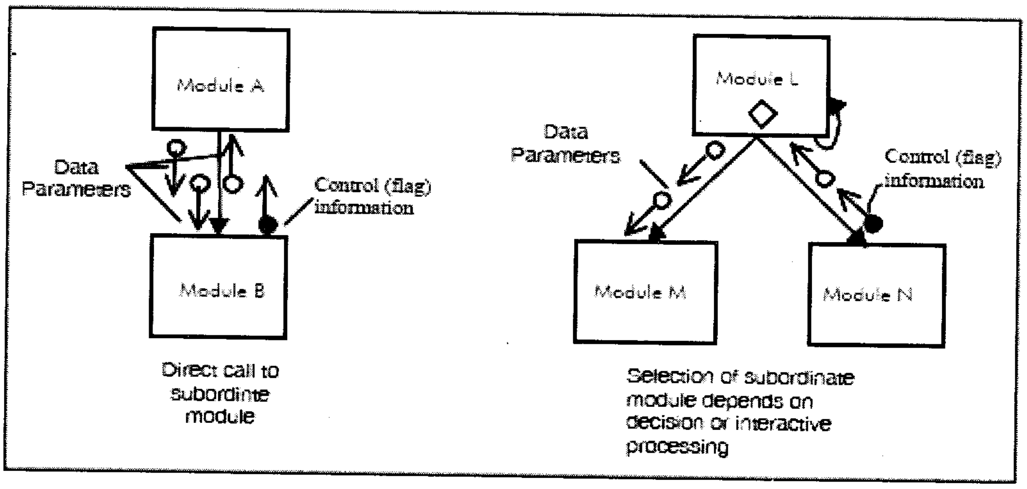

Module A
Data Parameters
Control (flag) information
Module B
Direct call to subordinte module
Module L
Data Parameters
Control (flag) information
Module M
Module N
Selection of subordinate module depends on decision or interactive processing

Figure 6.5: Structure Chart Example

6.6.2 Coupling

Coupling is a measure of interconnection among modules in a program structure. It may be represented on a spectrum as shown in Figure 6.6. Coupling depends on interface complexity between modules, the point at which entry or reference is made to a module and what data passes across the interface.

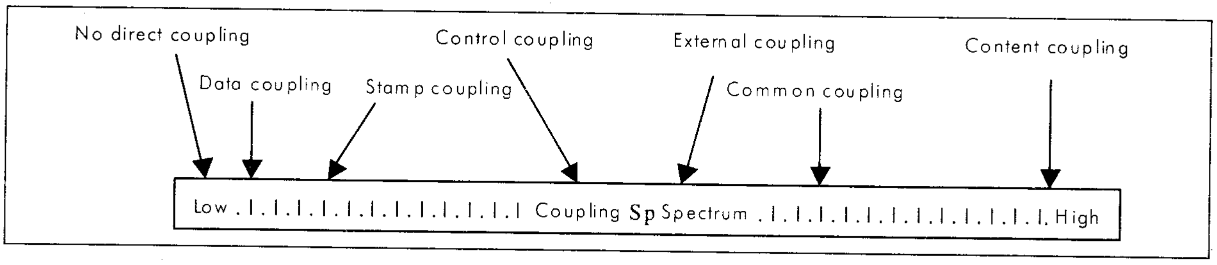

No direct coupling
Data coupling
Stamp coupling
Control coupling
External coupling
Common coupling
Content coupling
Low . I . I . I . I . I . I . I . I . I . I . I . I Coupling Sp Spectrum . I . I . I . I . I . I . I . I . I . I . I . I. High

Figure 6.6: Coupling Spectrum

In systems design, we strive for lowest possible coupling. Simple connectivity among modules result in software that is easier to understand and less prone to "ripple effect" caused when error occurs at one location and propogates through a system.

Figure 6.7 provides an example of different types of module coupling. Modules 'a' and 'd' are sub-routines to different modules. Each is unrelated and therefore no direct coupling occurs. Module 'c' is subordinate to module 'a' and is accessed via a conventional argument list through which data are passed; a one-to-one correspondence of items exists, low coupling (data coupling on the spectrum) is exhibited in this portion of structure. A variation of data coupling, called stamp coupling is found when a portion of a data structure (rather than simple arguments) is passed via a module interface. This occurs between modules b and a.

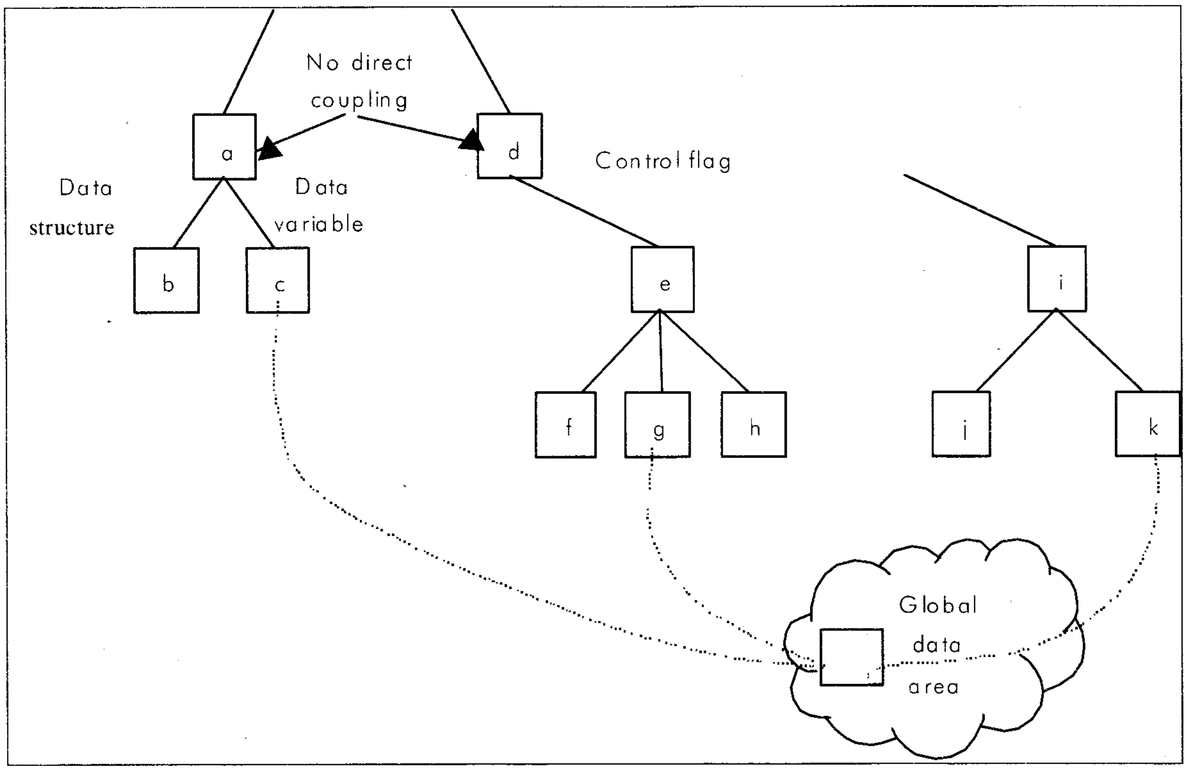


Figure 6.7: Types of Coupling

At moderate levels, coupling is characterized by passage of control between modules. Control coupling is very common in most software designs and is shown in Figure 6.7 where a control flag (a variable that controls decision in a subordinate or superordinate module) is passed between modules d and e.

Relatively high levels of coupling occur when modules are tied to an environment external to software. For example, I/O couples a module to specific devices, formats and communication protocols. External coupling is essential but should be limited to a small number of modules with a structure. High coupling also occurs when a number of modules refer to a global data area. Common coupling, as this mode is called, occurs among modules e.g., g and k, in which each uses a global data area.

The highest degree of coupling - content coupling - occurs when one module makes use of data or control information maintained within the boundaries of another module.

Secondly, content coupling occurs when branches are made into the middle of a module. This mode of coupling can be and should be avoided.

6.6.3 Cohesion

Cohesion is the degree to which a module performs a single task. A cohesive module performs a single task within a software procedure, requiring little interaction with procedures being performed in other parts of a program. Stated simply, a cohesive module should (ideally) do just one thing.

Cohesion may be represented as a spectrum shown in Figure 6.8. We always strive for high cohesion, although, the mid range of the spectrum is often acceptable. The scale for cohesion is non-linear. That is, low ended cohesiveness is much "worse" than the middle range which is nearly as "good" as high ended cohesion. In practice, a designer need not be concerned with categorizing cohesion in a specific module. Rather, the overall concept should be understood and low level of cohesion should be avoided when modules are designed.

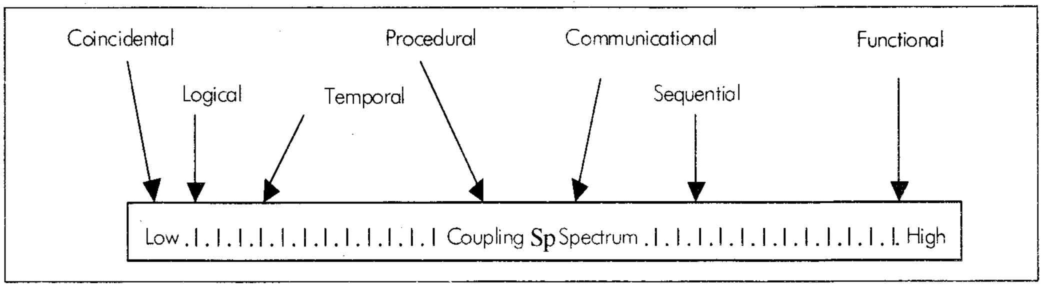

Coincidental
Logical
Temporal
Procedural
Communicational
Sequential
Functional
Low . I . I . I . I . I . I . I . I . I . I . I . I . I Coupling SpSpectrum . I . I . I . I . I . I . I . I . I . I . I . I. High

Figure 6.8: Spectrum of Cohesion

A module that performs a set of tasks that relate to each other loosely, if at all is termed coincidentally cohesive. A module that performs tasks that are logically related to each other (e.g., a module that produces all output regardless of type) is logical cohesiveness. When the module contain tasks that are related by the fact that, all must be executed within the same span of time, the module exhibits temporal cohesion.

As an example of low cohesion, consider a module that performs error processing for an engineering analysis package. The module is called, when computed data exceeds pre-specified bounds. It performs the following tasks:

- Computes supplementary data based on original computed data
- Produces an error report on user's workstation
- Performs follow-up calculations requested by user
- Updates a database
- Enables menu selection for subsequent processing

Although, the preceding tasks performed are loosely related, each is a separate function entity that might best be performed as separate modules. Combining the function in single module can only serve the likelihood of error propogation when a modification is made to any one of the processing tasks noted above.

Moderate levels of cohesion are relatively close to one another in the degree of module independence. When processing elements of a module are related and must be executed in a specific order, procedural cohesion exists. When all processing elements concentrate on one area of data structure, communicational cohesion is present. High cohesion is characterized by a module that performs one distinct procedural task.

As we have already noted, it is unnecessary to determine the precise level of cohesion. Rather, it is important to strive for high cohesion and recognize low cohesion so that software design can be modified to achieve greater functional independence.

6.6.4 Span of Control

Span of control refers to the number of subordinate modules controlled by a calling module. In general, we should seek to have no more than five to seven subordinate modules (Figure 6.9).

On the one hand, excessive span of control, meaning a high number of subordinate modules, create considerable overhead in determining which module to invoke under certain conditions and in establishing calling sequences to pass data and results. On the other hand, it typically results from not adhering to the coupling and cohesion objectives discussed previously.

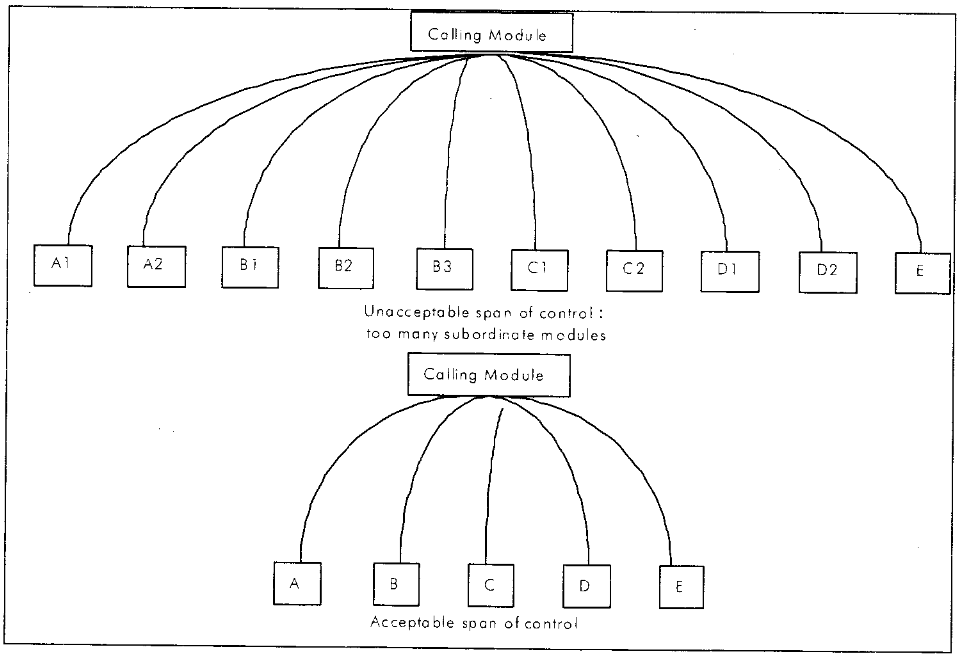

Calling Module
A1
A2
B1
B2
B3
C1
C2
D1
D2
E
Unacceptable span of control : too many subordinate modules
Calling Module
A
B
C
D
E
Acceptable span of control

Figure 6.9: Span of Control

6.6.5 Module Size

How large should a module be? While it is impossible to fix a specific number of instructions, there are useful guidelines to manage module size.

Some organizations have established rules to manage module size. A common one is that no module should contain more than 50 instructions. Another is that, the listing of source code for a module should fit into a single printed page. In some situations, these rules are appropriate but in others they result in arbitrary decisions that miss the point of managing module size.

In general, we should seek designs in which the modules focus on a single purpose, are highly cohesive and are loosely coupled. The size of a module also depends on the language. Lesser instructions are used in a 4GL, while in any 3GL, the number of instructions may be higher for the same task.

6.6.6 Shared Modules

Shared use results from the desire to have a certain function, calculation or type of processing performed in one place in a system. We want to design the system so that a certain calculation (such as determination of sales tax) is performed once. Then the module can be shared throughout the system by invoking it from various calling modules.

Why share modules? There are several reasons:

- Sharing modules minimizes the amount of software that must be written.
- It minimizes the number of changes that must be made during system maintenance.
- Having a single shared module reduces the risk of error.

Many systems establish library modules - predefined procedures - that are included in the system's program library. The routine is quickly invoked by a single command or call. Shared library modules are a means by which good designers follow the principles of system design.

6.6.7 Modular Specifications

Modular specifications are entered in a document, that is, the complete details of the modular design process of the system have been included into it. Modular specifications identify inputs becomes the outputs and describe the functional aspects of the system. The specification then are used as a basis for the graphic representation of the modular system. So we can say that, it serves as a blue print and helps in developing and implementing the new modular system.

This also forms the primary documentation on which the system, maintaining persons, will fall back upon after the system in use. Later on, this document is normally divided into different parts for easy reference. Thus, we get system manual, user manual, operational manual. Since the specifications are prepared as a plan, it becomes sometime necessary to modify it after taking into consideration the practical difficulties or bottlenecks or errors found during later stage of developments. Modular specification should include all the details necessary to implement the system and to understand the whole working of the system.

6.7 USER INTERFACE DESIGN

The input design is the link that ties the information system into the world of its users. Input specifications describe the manner in which data enters the system for processing.

Input design features can ensure the reliability of the system and produce results from accurate data, or they can result in the production of erroneous information.

Input design consists of developing specifications and procedures for data preparation steps necessary to put transaction data into a usable form for processing data entry, the activity of putting data into the computer for processing.

6.7.1 Objectives of Input Design

Five objectives guiding the design of input focus on:

- Controlling the amount of input required
- Avoiding delay
- Avoiding errors in data
- Avoiding extra steps
- Keeping the process simple

6.8 FORM DESIGN

We have learned that data provide the basis for information systems. Without data there is no system, but data must be provided in the right form for input and the information produced must be in a format acceptable to the user. In either case, it is still data-the basic element of a printed form.

6.8.1 What is a Form?

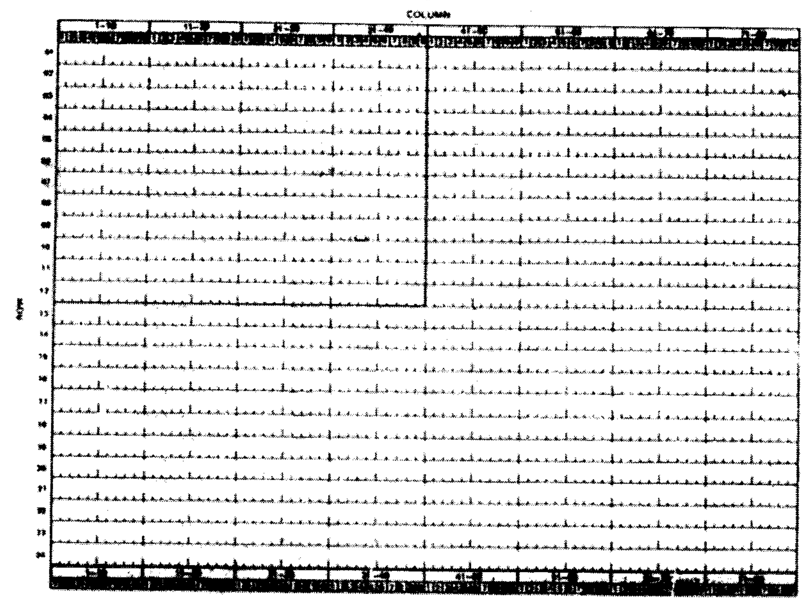
People read from forms, write on forms, and spend billions of hours handling forms and filing forms. The data the forms carry come from people, and the informational output of the system goes to people. So the form is a tool with a message; it is the physical carrier of data-of information. It also can constitute authority for action. For example, a purchase order says BUY, a customer's order says SHIP, and a paycheck says PAY TO THE ORDER OF. Each form is a request for action. It provides information for making decisions and improving operations.

With this in mind, it is hard to imagine a business operating without using forms. They are the vehicles for most communications and the blue-print for many activities. As important as a printed form is, however, the majority of forms are designed by poorly trained people. People are puzzled by confusing forms they ask for directions on how to read them and how to fill them out. When a form is poorly designed, it is a poor (and costly) administrative tool.

6.8.2 Classification of Forms

A printed form is generally classified by what it does in the system. There are three primary classifications: action, memory, and report forms. An action form requests the user to do something-get action. (Examples are purchase orders and shop orders. A memory form is a record of historical data that remains in a file, is used for reference, and serves as control on key details. (Examples are, inventory records, purchase records, and bond registers. A report form guides supervisors and other administrators in their activities. It provides data on a project or a job.

Display Layout Sheet



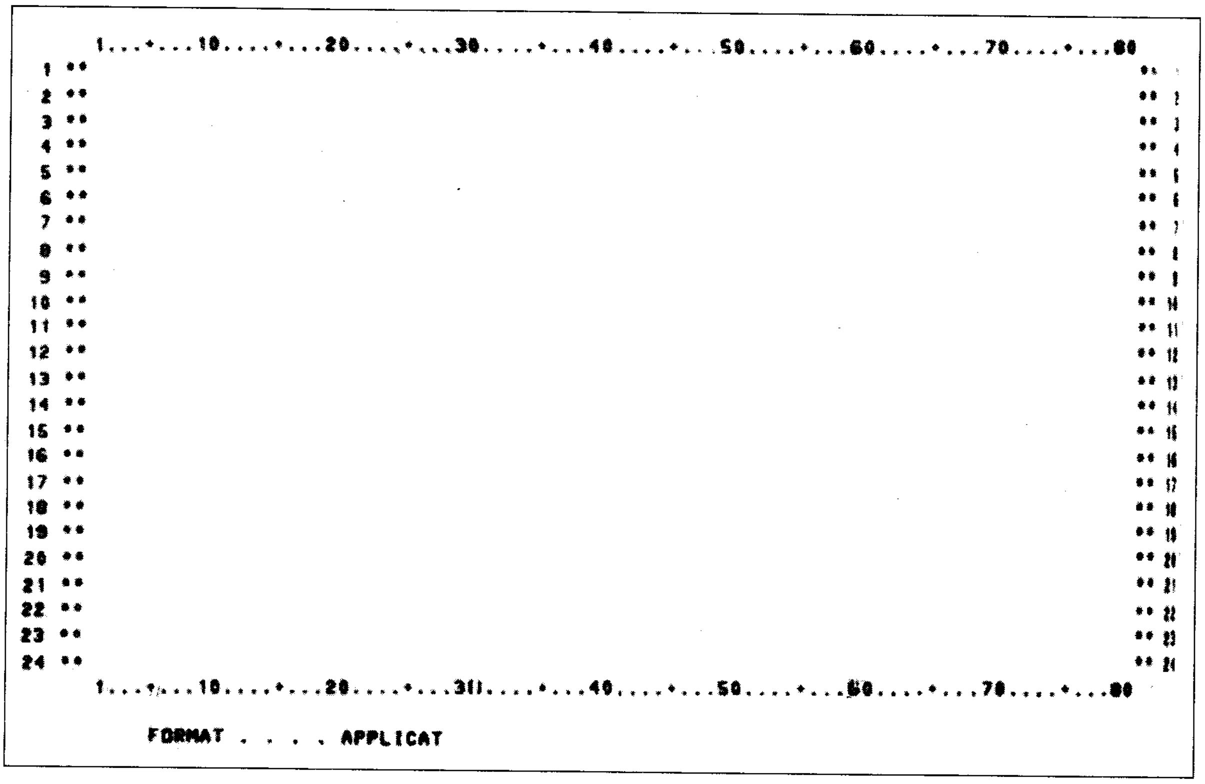


Figure 6.10: Different Types of Forms

6.8.3 Requirements of Forms Design

Forms design follows analyzing forms, evaluating present documents, and creating new or improved forms. Bear in mind that detailed analysis occurs only after the problem definition stage and the beginning of designing the candidate system. Since the purpose of a form is to communicate effectively through forms design, there are several major requirements:

1. ***Identification and Wording:*** The form title must clearly identify its purpose. Columns and rows should be labeled to avoid confusion. The form should also be identified by form name or code number to make it easy to reorder.

2. ***Maximum Readability and Use:*** The form must be easy to use and fill out. It should be legible, intelligible, and uncomplicated. Ample writing space must be provided for inserting data. This means analyzing for adequate space and balancing the overall form's layout, administration, and use.

3. ***Physical Factors:*** The form's composition, color, layout (margins, space, etc.), and paper stock should lend themselves to easy reading. Pages should be numbered when multipage reports are being generated for the user.

4. ***Order of data Items:*** The data requested should reflect a logical sequence. Related data should be in adjacent positions. Data copied from source documents should be in the same sequence on both forms. Much of this design takes place in the forms analysis phase.

5. ***Ease of Data Entry:*** If used for data entry, the form should have field positions indicated under each column of data and should have some indication of where decimal points are (use broken vertical lines).

6. ***Size and Arrangement:*** The form must be easily stored and filed. It should provide for signatures. Important items must be in a prominent location on the form.

7. ***Use of Instructions:*** The instructions that accompany a form should clearly show how it is used and handled.

8. ***Efficiency Considerations:*** The form must be cost effective. This means eliminating unnecessary data and facilitating reading lines across the form. To illustrate, if a poorly designed form causes 10 supervisors to waste 30 seconds each, then 5 minutes are lost because of the form. If (the firm uses 10,000 of these forms per year, then 833 hours of lost time could have been saved by a better forms design.

9. ***Type of Report:*** Forms design should also consider whether the content is executive summary, intermediate managerial information, or-supporting-data. The user requirements for each type often determine the final form design.

6.8.4 Form Control

The first step in tonus control is to determine whether a, form is necessary, Managing the hundreds of tonus in a typical organization requires a forms control program. Forms control is a procedure for (1) providing improved and effective forms (2) Reducing printing costs, and (3) Securing adequate stock at all times.

The first step in a procedure for forms control is to collect group, index, stock, and control the tonus of the organization. Each form is identified and' classified by the function it pelf onus and whether it is a flat form, a snap-out form, or something else. Once classified, a form is evaluated by the data it requires, where it is used, and how much it overlaps with other forms. The object is to get rid of unnecessary tonus and improve those forms that are necessary.

Before launching a tonus control program, the designer needs to consider several questions:

1. Who will be responsible for improving tonus design?
2. Should forms be produced in house or assigned to an outside printer?
3. What quantity should be printed? What is the break-even point on printing forms?
4. How much lead time is required to replenish forms?
5. How will one handle reorders? Who will initiate them? In what way?
6. How will obsolete forms be handled?
7. What should be the life of the form?
8. Where and how should the forms be stored?

If questions of this nature are not addressed in advance, the organization is probably not ready to launch a tonus control program.

6.9 INPUT STAGES

Several activities have to be carried out as part of the overall input process. They include some or all of the following:

- Data recording (i.e., collection of data at its source):
- Data transcription (i.e., transfer of data to an input form);
- Data conversion (i.e., conversion of the input data to a computer acceptable medium);
- Data verification (i.e., checking the conversion);
- Data control (i.e., checking the accuracy and controlling the flow of the data to the computer);
- Data transmission (i.e., transmitting or transporting the data to the computer);
- Data validation (i.e., checking the input data by a program when it enters the computer system)
- Data correction (i.e., correcting the errors that are found in any of the earlier stages).

6.9.1 Input Media

Once the input types and their contents have been examined the analyst can start to think about input devices of which there is a very wide range.

Much careful thought has to be given to the choice of the input media and devices. Consideration can be given to:

- Type of input
- Flexibility of format
- Speed
- They-accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Off-line facilities
- Need for specialized documentation
- Storage and handling requirements
- Automatic features
- Hard copy requirements
- Security
- Ease of use
- Environment of data capture
- Portability

- Compatibility with other systems
- Cost

6.9.2 Avoiding Errors in Data

Every effort must be made to ensure that input data remains accurate from the stage at which it is recorded and documented to the stage at which it is accepted by the computer. This can only be achieved by careful control each time data is handled.

The rate at which errors occur depends on the quantity of data, since the smaller the amount of data to input, the fewer the opportunities for errors.

Poor form design can lead to a

- Misunderstanding of the instructions
- Insufficient space to write clearly.

The effectiveness of checking data by verification or sight-checking can only be assessed by keeping individual records of the preparations of the input data and 'tracing' errors which are subsequently found by the computer or even later in the system, back to their source.

6.9.3 CRT Screen Design

Many online data entry devices are CRT screens that provide instant visual verification of input data and a means of prompting the operator. The operator can make any changes desired before the data go to the system for processing. A CRT screen is actually a display station that has a buffer (memory) for storing data. A common size display is 24 rows of 80 characters each or 1,920 characters.

There are two approaches to designing data on CRT screens: manual and software utility methods. The manual method uses a worksheet much like a print layout chart. The menu or data to be displayed are blocked out in the areas reserved on the chart and then they are incorporated into the system to formalize data entry. For example, we use dBASE II software commands to display a menu on the screen. The first command in the partial program is interpreted by the, system as follows: "Go to row 10 and column 10 on the screen and display (SAY) the statement typed between quotes. The same applies to the next three commands. The command "WAIT TO A" tells the system to keep the menu on the screen until the operator types the option next to the word "WAITING".

The main objective of screen display design is simplicity for accurate and quick data capture or entry. Other guidelines are:

1. Use the same format throughout the project.
2. Allow ample space for the data. Overcrowding causes eye strain and may tax the interest of the user.
3. Use easy-to-learn and consistent terms, such as "add," "delete," and "create."
4. Provide help or tutorial for technical terms or procedures.

The second approach to designing screen layouts is through software utility, usually provided by the CRT vendor. For example, IBM provides a Screen Design Aid (SDA) package that allows the designer (at the terminal) to modify the display components.

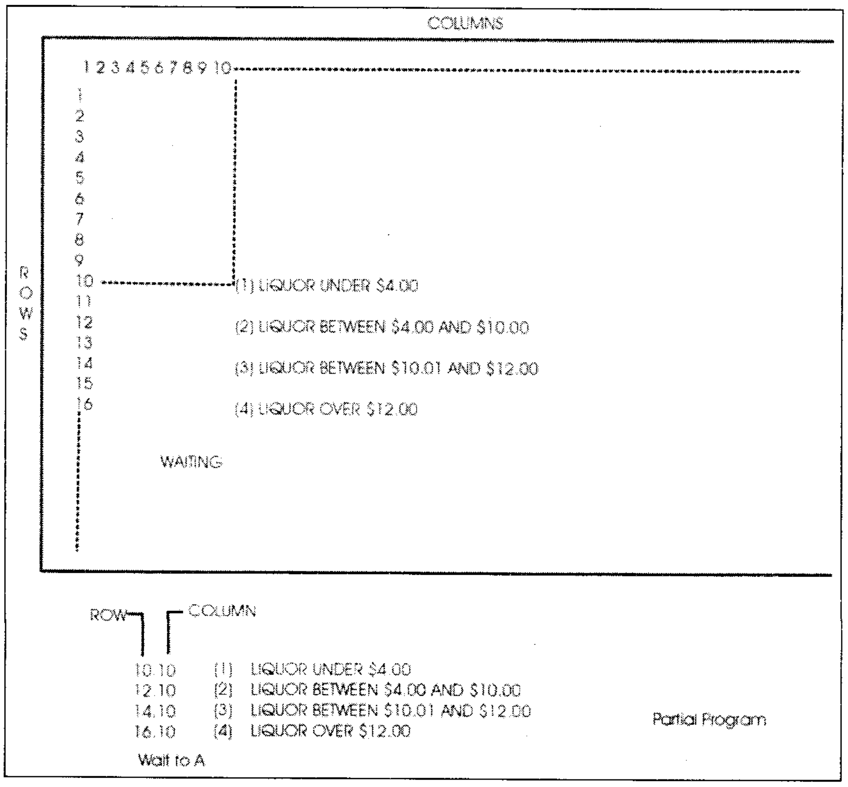

COLUMNS
1 2 3 4 5 6 7 8 9 10
ROWS
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(1) LIQUOR UNDER $4.00
(2) LIQUOR BETWEEN $4.00 AND $10.00
(3) LIQUOR BETWEEN $10.01 AND $12.00
(4) LIQUOR OVER $12.00
WAITING
ROW
COLUMN
10.10 (1) LIQUOR UNDER $4.00
12.10 (2) LIQUOR BETWEEN $4.00 AND $10.00
14.10 (3) LIQUOR BETWEEN $10.01 AND $12.00
16.10 (4) LIQUOR OVER $12.00
Wait to A
Partial Program

Figure 6.11: The CRT Design

In online applications, information is displayed on the screen. The layout sheet for displayed output is similar to the layout chart used for designing input. Areas for displaying the information are blocked out, leaving the rest of the screen blank or for system status information. Allowing the user to review sample screens can be extremely important because the user is the ultimate judge of the quality of the output and, in turn, the success (or failure) of the system. For example, the following shows editing output for a student birth date.

Display: DATE OF BIRTH (mm/dd/yy) 23/19/80

RESPONSE: MONTH EXCEEDS 12

SUGGESTS A RETRY: DATE OF BIRTH (mm/dd/yy)

6.10 FILE DESIGN

The first requirement of physical design of the system is to decide, how the logical data structures which have been defined in the logical system definition are to be physically stored on the backing storage device in the form of files.

Files are the heart of computer application. Constructing, files we must understand the basic types of files used to describe the file hierarchy.

6.10.1 Types of Files

There are various types of Files in which the records are collected and maintained. They are categorized as:

- Master File
- Transaction File
- Table File
- Report File
- Back-up File
- Archival File
- Dump File
- Library File

Master

Master files are the most important type of files. Most file design activities concentrate here. In a business application, these are considered to be very significant because they contain the essential records for maintenance of the organization's business. A master file can be further categorized. It may be called as a reference master file, in which the records are static or unlikely to change frequently. For example, a product file containing descriptions and codes, a customer file containing name, address and account number, are example of reference files-Alternatively, it may be described as a dynamic master file. In this file, we keep records which are frequently changed (updated) as a result of transactions or other events. These two types of master file may be kept as separate files or may be combined, for example, a sales ledger file containing reference data, such as name, address, account number, together with current transaction and balance outstanding for each customer.

Transaction

A transaction is a temporary file used for two purposes. First of all, it is used to accumulate data about events as they occur. Secondly, it helps in updating master files to reflect the result of current transactions. The term transaction refers to any business event that affects the organization and about which data is captured. Examples of common transactions in the organization are, making purchases, hiring of workers and recording of sales.

Table

A special type of master file is included in many systems to meet specific requirements, where data must be referenced repeatedly. Table files are permanent files containing reference data used in processing transactions, updating master file or producing output. As the name implies, these files store reference data in tabular form. Table files conserve memory space and make the program maintenance easier by storing data in a file, that otherwise would be included in programs or master file records.

Report

Report files are collected contents of individual output reports or documents produced by the system. They are created by the system, where many reports are produced by the system but the printer may not be available for all the reports. This situation frequently arises, when the computer carry out three functions - input, processing and output simultaneously, rather than executing each function in sequence. In this case, the computer writes the report contents to a file on a magnetic tape or disk, where it remains until it can be printed. That file is called the report file which contains the unprinted output data. The process of creating it is known as spooling which means that, output cannot be printed when it is produced but is spooled into a report file. Then, depending on the availability of printer, the system will be instructed to read the report file and print the output on the printer.

Backup

It is a copy of master, transaction or table file that is made to ensure a copy is available if anything happens to the original.

Archival

These files are copies made for long term storage of data that may be required at a much later date. Usually, archival files are stored far away from the computer centre so that they cannot be easily retrieved for use.

Dump

This is a copy of computer-held data at a particular point of time. This may be a copy of master file to be retained to help recovery in the event of a possible corruption of the master file or it may be part of a program in which error is being traced.

Library

Library file generally contains application programs, utility programs and system software packages.

6.10.2 File Structure

To learn about files, we need to understand the basic terms used to describe the file hierarchy. The terms we will cover are byte, data field, record, file and database (see Figure 6.12).

- *Byte:* A byte is an arbitrary set of eight bits that represents a character. It is the smallest addressable unit in today's computers.
- *Field:* One or more bytes are combined into a data item to describe an attribute of an object. For example, if the object is an employee, one attribute name may be sex, name, age or designation. A field is sometimes referred to as data item or element. It is actually a physical space on tape or disk, whereas, a data item is the data stored in the field.
- *Record:* A data item related to an object is combined into a record. A hospital patient has a record with his/her name, address, age, sex, unique key or ID number. The patient's tag number or a unique number could be used as an identifier for processing the record.

In record design, we distinguish between logical and physical records. A logical record maintains a logical relationship among all the data items in the record. It is the way a program or user sees the data. In contrast, a physical record is the way data are recorded on a storage medium. To illustrate, Figure 6.13 shows a programmer who requires a five record file in a particular sequence (A, D, C, B, S). The programmer does not know about the physical map on the disk. The software presents the logical records in the required sequence. This capability is unique to database design.

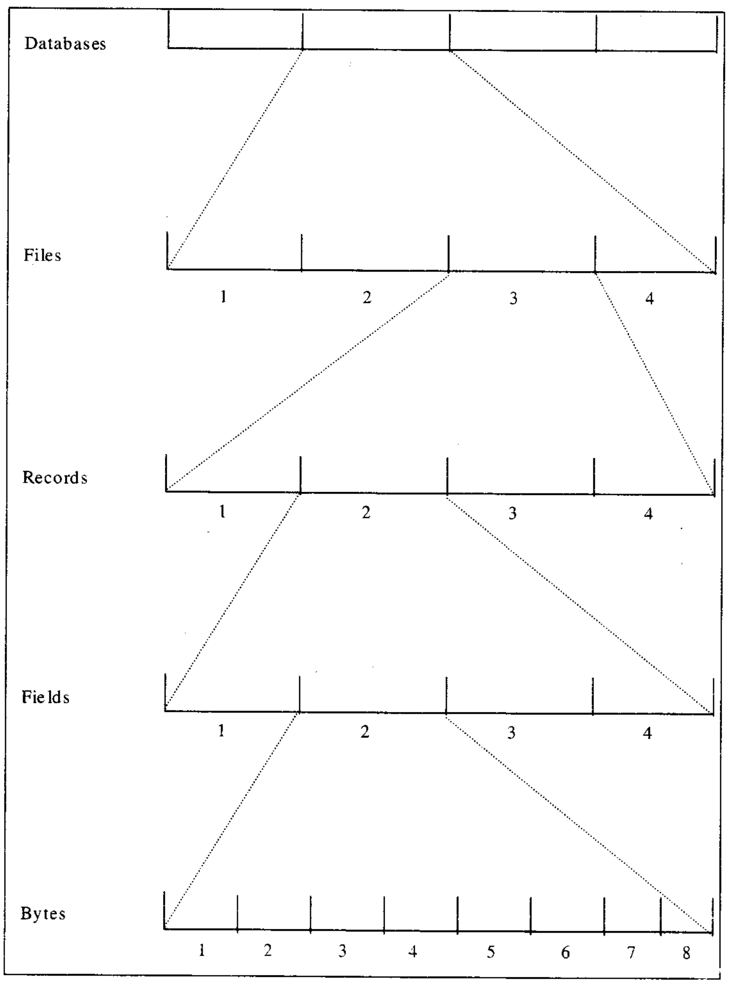
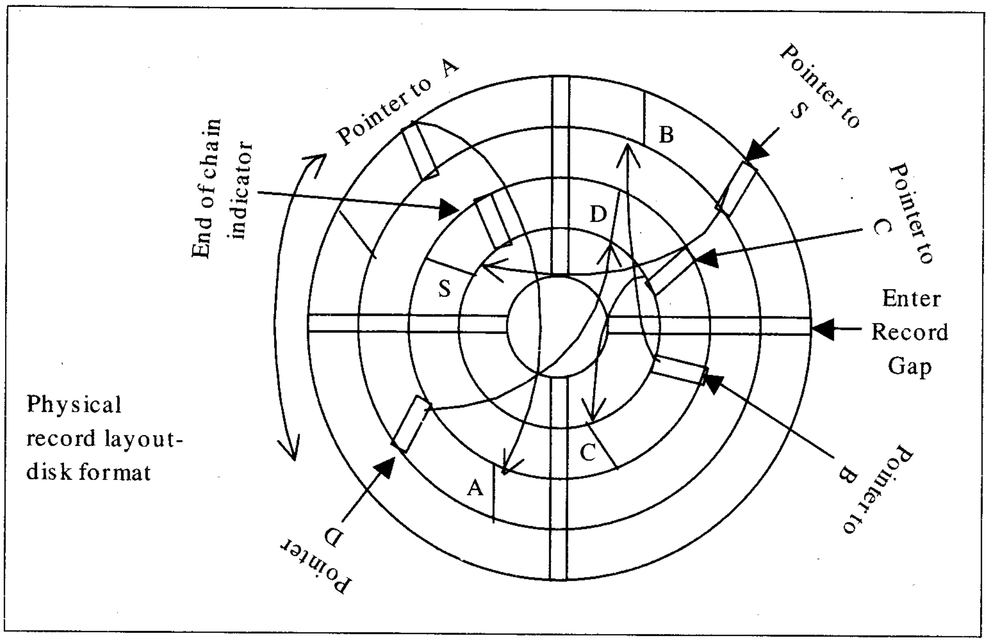


Figure 6.12: Hierarchy of Files



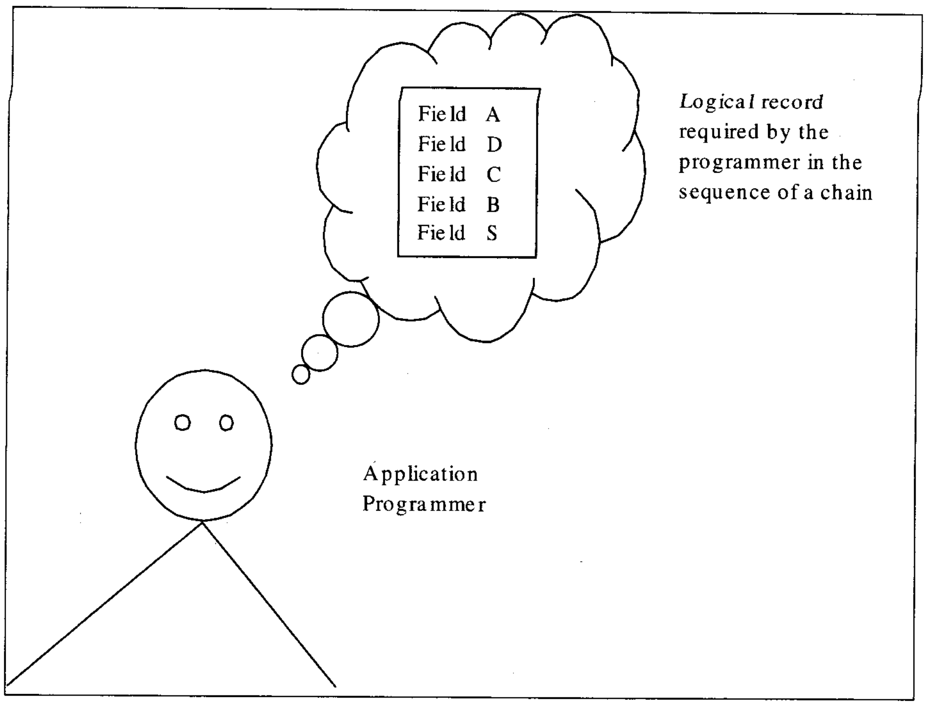

Field A
Field D
Field C
Field B
Field S
Logical record required by the programmer in the sequence of a chain
Application Programmer

Figure 6.13: Physical and Logical Records - A Contrast

- ***File:*** A collection of related records make up a file. The size of a file is limited by the size of memory or the storage medium. Two characteristics determine how files are organized: activity and volatility. File activity specifies the percentage of actual records processed in a single run. If a small percentage of records is accessed at any given time, the file should be organized on disk for direct access. In contrast, if a fair percentage of records is affected regularly, then sorting the file on tape would be more efficient and less costly. File volatility addresses the properties of record changes. File records with substantial changes are highly volatile, meaning the disk design would be more efficient than tape. Think of the airline reservation system and the high volatility through cancellations, additions, and other transactions, compared to traditional payroll, which is relatively dormant. Higher the volatility, more attractive is disk design.
- ***Database:*** The highest level in the hierarchy is the database. It is a set of interrelated files for real time processing. It contains the necessary data for problem solving and can be used by several users accessing data concurrently.

6.10.3 File Organization

A file is organized to ensure that records are available for processing. As mentioned earlier, it should be designed in line with the activity and volatility of the information and the nature of the storage media and devices. Other considerations are:

- Cost of file media,
- Inquiry requirements, and
- File privacy, integrity, security and confidentiality.

There are four methods of organizing files: sequential, indexed sequential, inverted list and direct access. Each method is explained.

Sequential

Sequential organization simply means storing and sorting in physical contiguous blocks within files on tape or disk. Records are also in sequence within each block. To access a record, design is best suited for "get next" activities, reading one record after another without a search delay.

In sequential organization, records can only be added at the end of the file. It is not possible to insert a record in the middle of the file without rewriting the file.

Indexed Sequential

Like sequential organization, keyed sequential organization stores data in physically contiguous blocks. The difference is in the use of indexes to locate records. To understand this method, we need to distinguish among three areas in disk storage: prime area, overflow area and index area. The prime area contains file records stored by key or ID number. All records are initially stored in the prime area. The overflow area contains records added to the file that cannot be placed in logical sequence in the prime area. The index area is more like the data dictionary. It contains keys of records and their locations on the disk. A pointer associated with each key is an address that tells the system, where to find a record.

Indexed sequential organization reduces the magnitude of the sequential search and provides quick access for sequential and direct processing. The primary drawback is extra space required for the index. It also takes longer to search the index for data access or retrieval.

Chaining and Inverted List Organization

Like the indexed sequential storage method, the inverted list organization maintains an index. The two methods differ, however, in the index level and record storage. The indexed sequential method has a multiple index for a given key, whereas, the inverted list method has a single index for each key type. In an inverted list, records are not necessarily stored in a particular sequence. They are placed in the data storage area but indexes are updated for the record keys and location.

The inverted lists are best for applications that request specific data on multiple keys. They are ideal for static files because additions and deletions cause expensive pointer updating.

Direct Access Organization

In direct access file organization, records are placed randomly throughout the file. Records need not be in sequence because they are updated directly and rewritten back in the same location. New records are added at the end of the file or inserted in specific locations based on software commands.

Records are accessed by addresses that specify their disk locations. An address is required for locating a record, for linking records or for establishing relationships. Addresses are of two types: absolute or relative. An absolute address represents the physical location of the record. It is usually stated in the format of sector/track/record number. For example, 3/14/6 means, go to sector 3, track 14 of that sector and the sixth record of the track. One problem with absolute address is that they become invalid when the file that contains the records is relocated on the disk. One way around this is to use pointers for the updated record.

A relative address gives a record location relative to the beginning of the file. There must be fixed length records for reference. Another way of locating a record is by the number of bytes it is from the beginning of the file. Unlike absolute addressing, if the file is moved, pointers need not be updated because the relative location of the record remains the same. Advantages and disadvantages of all the above discussed file organizations are briefly discussed in Figure 6.14.

Method	Advantage	Disadvantage
Sequential	Simple to design Easy to program Variable length and blocked records available Best use of storage space	Records cannot be added to middle of file
Indexed sequential	Records can be inserted or updated in middle of file Processing may be carried out sequentially or randomly	Unique keys required Processing occasionally slow Periodic reorganization of file required
Inverted list	Used in applications requesting specific data on multiple keys	
Random	Records can be inserted or updated in middle of file Better control over record location	Calculating address required for processing Variable length records nearly impossible to process

Figure 6.14: File Organization, A Summary

6.10.4 File Design Considerations

The choice of the most suitable data organization for a particular group of applications is important, perhaps even crucial, to the eventual successful performance of systems which use the data. It is, however, possible to draw up a series of alternatives in designing files and to evaluate those alternatives against given criteria. The considerations may be as follows:

6.11 DATABASE DESIGN

Sometime ago, the database was unique to large corporations with mainframes. Today, it is recognized as a standard of MIS and is available for virtually, every size of computer. Before the database concept became operational, users had programs that handled their own data independent of other users.

It was a conventional file environment with no data integration or sharing of common data across applications. In a database environment, common data are available and used by several users. Instead of each program (or user) managing its own data, data across applications are shared by authorized users with the database software managing the data as an entity. A program now requests data through the Database Management System (DBMS) which determines data sharing.

The general theme behind a database is to handle information as an integrated whole. There is none of the artificiality that is normally embedded in separate files or applications. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In database design, several specific objectives are considered.

- *Controlled Redundancy:* Redundant data occupies space and, therefore is wasteful. If versions of the same data are in different phases of updating, the system often gives conflicting information. A unique aspect of database design is storing data only once. This controls redundancy and improves system performance:
- *Ease of Learning and Use:* A major feature of a user friendly database package is, how easy it is to learn and use. Related to this point is that a database can be modified without interfering with established ways of using the data.
- *Data Independence:* An important database objective is changing hardware and storage procedure or adding new data without having to write application programs. The database should be tunable to improve performance without rewriting programs.
- *More Information at low Cost:* Using, storing and modifying data at low costs is important. Although hardware prices are falling, software and programming costs are on the rise. This means that programming and software enhancement should be kept simple and easy to update.
- *Accuracy and Integrity:* The accuracy of a database ensures that data quality and content remains constant. Integrity controls detect data inaccuracies where they occur.
- *Recovery from Failure:* With multiuser access to a database, the system must recover quickly after it is down with no loss of transaction. This objective also helps maintain data accuracy and integrity.
- *Privacy and Security:* For data to remain private, security measures must be taken to prevent unauthorized access. Database security must be positively identified and their action monitored.
- *Performance:* This objective emphasizes response time to enquiries suitable to the case of data. How satisfactory the response time is, depends on the nature of the user database dialog.

In addition to the database itself, a set of programs is necessary to facilitate adding new data as well as modifying and retrieving existing data within a database. This set of programs is referred to as a Database Management System (DBMS). A database system merge data into one pool shared by all systems so that any change automatically affects all relevant systems. The following figures define the difference between the traditional file systems and database management system.

Figure 6.15 shows the Traditional File Systems in which each system is responsible for its own data.

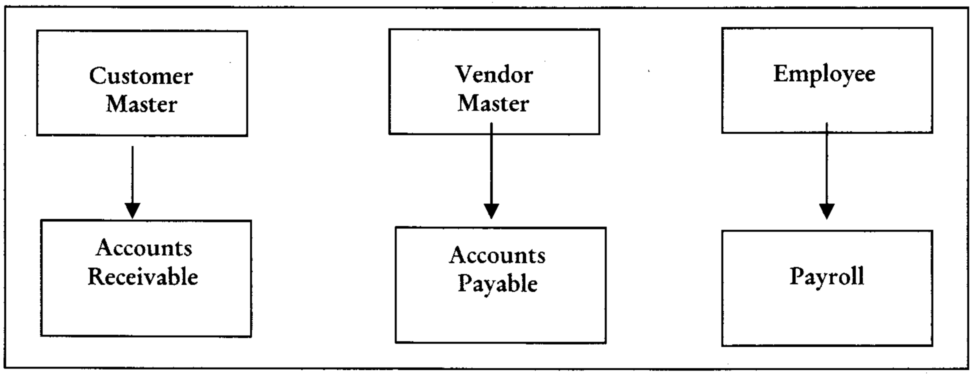


Figure 6.15: Traditional File System

Figure 6.16 shows the Database Management Systems in which data is centralized.

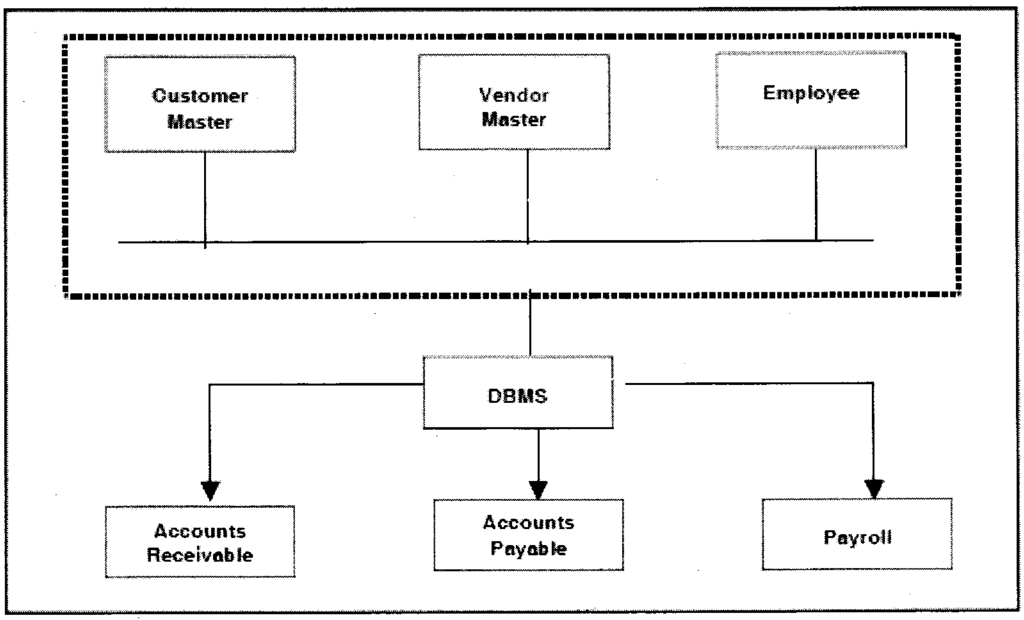


Figure 6.16: Data Base Management System

Specific advantages of database are:

1. ***File Consolidation:*** Pooling data reduces redundancy and inconsistency and promotes cooperation among different users. Since databases link records together logically, a data change in one system will cascade through all the other system using the data.
2. ***Program and File Independence:*** This feature separates the definition of the files from their programs, allowing a programmer to concentrate on the logic of the program instead of precisely how to store and retrieve data.
3. ***Access Versatility:*** Users can retrieve data in many ways. They enjoy the best of both worlds - sequential access for reporting data in a prescribed order and random access for rapid retrieval of a specific record.
4. ***Data Security:*** Usually a DBMS includes a password system that controls access to sensitive data. By limiting their access to read-only, write-only, or specified records, or even fields in records, passwords can prevent certain users from retrieving unauthorized data.
5. ***Program Development:*** Programmers must use standard names for data items rather than invent their own from program to program. This allows the programmer to focus on desired function.
6. ***Program Maintenance:*** Changes and repairs to a system are relatively easy.
7. ***Special Information:*** Special-purpose report generators can produce reports with minimum effort.

6.11.1 Logical and Physical Views of Data

In database design, several views of data must be considered along with the persons who use them. In addition to data structuring, where relationships are reflected between and within entities, we need to identify the application program's logical views of data within an overall logical data structure. The logical view is what the data looks like, regardless of how they are stored. The physical view is the way data exist in physical storage. It deals with, how data are stored, accessed, or related to other data in storage. There are four views of data, out of which, three are logical and one is physical. The logical views are the user's view, the programmer's view and the overall logical view, called a SCHEMA.

SCHEMA

Once a database system has been designed, it will be possible to identify each type of data item, data aggregate, record and set by a name or code. It will be possible to state which data item types go together to make data aggregate types and record types, and to identify which record types are members and owners of set types. A coded set of tables describing this information and stored in the computer system on direct access devices is called a SCHEMA. It is a description of the data structure which is separate from the data itself. The schema describes the areas, their identifiers and page sizes, and indicates how these are related to the records and sets. In other systems, a different set of tables is used for this.

The SCHEMA, therefore is the view of the data, the overall logical data structure which is held by the DBMS. Each time a program requires data, the DBMS will look up in the SCHEMA for the details of the structure of the data requested. For example, if the program requires an occurrence of a set, the DBMS will look up in the SCHEMA, which record types are required, how to find the relevant records given a certain key by the program, and perhaps, also, which areas the pages containing the relevant data are stored in.

SUB-SCHEMA

In a database system, it is not always possible to allow programmers to write the data division of their choice for reasons of security or control. It is more useful to provide the programmer with a standard description of the logical data to be used in a particular application. All references to data within the program will be for this description, which is called a SUB-SCHEMA and is similar to the SCHEMA in structure. The DBMS has the job of matching data requests on a SUB-SCHEMA and data requests based on the schema.

6.11.2 Types of Database

In conventional file systems, groups of bytes constitute a field, one or more fields make a record, and two or more records make a file. In a database environment, a group of bytes constitutes a data item or segment, a collection of segments a data entry, and a series of data entries a data set. The complete collection of data sets is the database itself. With traditional processing of files, records are not automatically related, so a programmer must be concerned with record relationships. Often the Files are stored and processed by record keys, just as we sorted the transaction file. Databases relate data sets in one of three models: hierarchical, network, or relational.

Hierarchical Model

In a hierarchical structure, sometimes referred to as a tree structure, the stored data get more and more detailed, as one branches further and further out on the tree. Each segment, or node, may be sub-divided into two or more subordinate nodes, which can be further sub-divided into two or more additional nodes. However, each node can have only one "parent" from which it emanates. The Figure 6.17 shows the hierarchical structure.

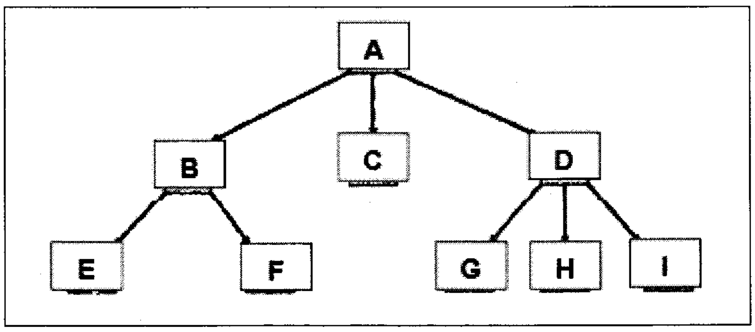


Figure 6.17: Hierarchical Structure

Network Model

The network related data sets are similar to hierarchical ones, except that a node may have more than one parent. Thus, a hierarchical DBMS is a subset of network DBMS. The trade off between the simplicity of design of a hierarchical structure and the storage efficiency of a network structure is a very important consideration in database implementation. The Figure 6.18 shows the network structure.

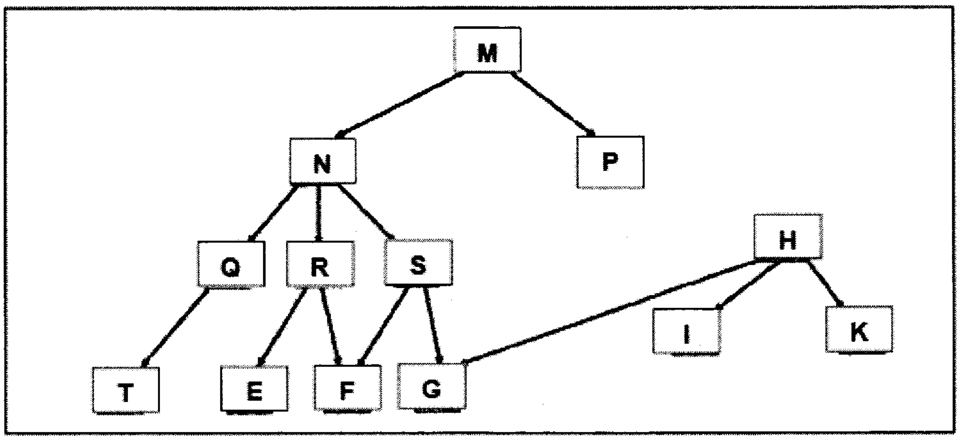


Figure 6.18: Network Structure

Relational Model

The relational structure, however, organizes the data in terms of two dimensional tables, that is, relational data sets order data in a table of rows and columns and differ markedly from their hierarchical or network counterparts. "There are no parent or node data sets as shown in Figure 6.19. In a relational database management systems, we have the same concept of files, records, and fields. Files are represented by two-dimensional tables, each of which is called a "relation". Records, which can be visualized as rows in the table, are called "Tuples". Fields can be visualized as columns, and are called by attribute names, or domains.

For example, note that, in the supplier table in Figure 6.19 we have three Tuples, or rows, and three attribute names or columns. If we need to know the name of the supplier of blue chairs, the relational DBMS searches the type and color columns of the furniture, table and finds supplier number 30, and then, it scans the supplier table for number 30, which turns out to be PANKAJ'S. Since each "record" is a row in the table and each "field" a column, an inventory system of 1600 Tuples, each with 5 attributes, would create a table of 1600 rows and 5 columns.

Product Number	Type	Color	Quantity in stock	Supplier Number
2589	Table	White	4	26
2892	Chair	Blue	6	30
3471	Chair	Light Green	20	133
3678	Desk	Brown	9	150
3689	Stool	Brown	25	159

SUPPLIER

Supplier Number	Supplier Name	Amount of Purchases This year
30	PANKAJ	26,035.00
26	DINESH	13,960.00
159	RAJESH	75,286.00

Figure 6.19: Relational Structure

A relational DBMS can perform the following basic operations:

- Create or delete tables
- Update, insert, or delete rows
- Add or delete columns
- Copy data from one table into another
- Retrieve or query a table, row or column
- Print, recognize, or read a table or row
- Join or combine tables based on a value in a table.

Since the relational structure organizes the data in terms of two dimensional tables, they offer great flexibility and a high degree of data security. The relational structure uses relatively little memory or secondary storage. Unfortunately, the process of creating the tables is a rather elaborate procedure. Another disadvantage of this structure is that it generally requires more time to access information than either of the other two structures. This is because much more information must be searched in order to answer queries posed to the systems. In addition, some implementation use a fixed amount of storage for each field, resulting in insufficient storage utilization. In spite of these disadvantages, the relational structure has gained rapid acceptance and is currently the most popular of the three structures. Many experts predict that it will eventually replace the others completely.

6.11.3 Levels of Data Independence

The data independence may be of the following two levels:

(a) ***Physical Data Independence:*** If the data is designed in such a way that the physical storage techniques of the data can be changed without changing the application programs then this level of data independence is called as physical data independence.

(b) ***Logical Data Independence:*** If the database is organized in such a way that the logical structure can be changed without changing the application program, then, this level of data independence is called as logical data independence.

Reasons for Data Independence

Why the data independence is required? It is mainly due to the following two reasons:

(1) The development of one application is generally very expensive and time consuming. Therefore, the application should be developed considering the possibility of changes required in specifications of reports or queries in future. Hence, if the database is application independent, then there would not be any need to modify the application programs in future and hence will save time and money of the organization.

(2) The requirement of queries or reports may be changed in future depending upon different applications. If the database is application independent, then the data can be stored in one particular representation (e.g. say in binary) but can be viewed in another representation (say in decimal). Sometimes the units of numeric data can also be changed in future (say from centimeters to inches). So, there are a lot of possibilities of modification of specifications given by organization in future. Therefore, the data independence concept is ideal for such applications.

Check Your Progress

1. Fill in the blanks:
   (a) Abstraction is a concept related to ____________________.
   (b) ______________ is the strength of interconnections between modules.
   (c) ______________ design is the most popular methodology for developing system designs.
   (d) An entity having information together with operations that can be performed on it is called an ___________.
2. State true or false:
   (a) A good design should not have any consistency.
   (b) A good design should have loosely coupled and highly cohesive modules.
   (c) Support to top management in designing the system is an external constraint on MIS design.
   (d) In a modular system, each component must support abstraction.

6.12 LET US SUM UP

After analysis phase, the analyst focuses on the design of the system. The main objective of the design phase is to produce a model of the system. The best possible design should be correct, complete, efficient, consistent and maintainable. The basic principles of system design include problem partitioning, abstraction, top-down design, bottom-up design and modularity.

Design constraints can be internal or external to the system. The processing options available to the designers include Batch processing, Real-time processing, On-line processing or a combination of all.

Structured design is the most popular methodology for developing system design through a number of steps. Object-oriented design is the latest method of designing the real life systems. In structured design, a program is segmented into small, independent modules to minimize the complexity of the problem.

A module may be categorized as a sequential module, incremental module or parallel module, within the program structure. The concept of functional independence is a direct outgrowth of modularity and the concept of abstraction and information hiding. Cohesion is natural extension of the information hiding concept. Coupling is a measure of interconnection among in a program structue.

System design process include output design, Input design, file design, procedure design and control design. The result of the system design process is a document known as "system specification".

At last, prototyping is discussed, which is the process of creating, developing and refining a working model of the final operational system.

6.13 KEYWORDS

System Design: A bridge between analysis and development of software, which produces a model of the system.

Problem Partitioning: Breaking down of a complex problem into small modules that could be solved separately.

Top-down Design: A design that starts from the highest level component of the hierarchy and decomposes it into lower levels.

Bottom-up Design: A design that starts from the lowest level component of hierarchy and proceeds to higher level component.

Coupling: The strength of interconnections between modules.

Cohesion: The strength of different elements within a module.

Structured Design: The most popular methodology for developing system designs wherein a program is segmented into small, independent modules that are arranged in hierarchy and organized in top-down manner.

Modularization: Division of a complex program into small, independent segments/modules.

Sequential Module: A module that is referenced and executed without apparent interruption by the application software.

Incremental Module: A module that can be interrupted prior to completion by application software and subsequently, restarted at the point of interruption.

Parallel Module: A module that executes simultaneously with another module in concurrent multiprocessor environments.

Modula Module: Program components that combine data abstractons and procedured elements in an object-oriented manner.

6.14 QUESTIONS FOR DISCUSSION

1. What is the need of System Control? Explain.
2. Discuss various constraints under which the designer works.
3. What is structured design? Discuss.
4. Differentiate between:
   (a) Physical design and logical design.
   (b) Coupling and cohesion.
5. What are different principles of system design?
6. What are the different design objectives?
7. "Security is necessary for output as for input" Justify with example.
8. How will you design a good CRT screen?
9. Describe the designing process.
10. Write short notes on network and relational model.

Check Your Progress: Model Answers

1. (a) Problem partitioning
   (b) Coupling
   (c) Structured
   (d) Object
2. (a) False
   (b) True
   (c) False
   (d) True

6.15 SUGGESTED READINGS

S.A. Kelkar, *Structured System Analysis and Design*, Prentice Hall of India.

D R Jeffery; M J Lawrence, [Sydney, N.S.W. ;Englewood Cliffs, N.J.], *Systems Analysis and Design*, Prentice-Hall of Australia, 1985, ©1984.

James C Wetherbe, St. Paul , *Systems Analysis and Design*, West Pub. Co., ©1988.

UNIT IV

LESSON

7

SYSTEM IMPLEMENTATION

CONTENTS

7.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Define implementation and its types
- Describe the process of implementation
- Discuss review plan
- Describe maintenance.

7.1 INTRODUCTION

A crucial phase in the system life cycle is the successful implementation of the new system design. Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be completely new, replacing an existing manual or automated system or it may be major modification to an existing system. In either case, proper implementation becomes necessary so that a reliable system based on the requirements of the organisation can be provided. Successful implementation may not guarantee improvement in the organisation using the new system, but improper installation will prevent it. It has been observed that even the best system cannot show good result if the analysts managing the implementation, do not attend to every important details. This is an area where the systems analysts need to work with utmost care.

A design is a non-functional description of a solution. Though it is described (using design tools) in a manner in which its applicability, advantages and limitations can be easily examined, it is no way a working system. It is like the blueprint of a building construction – a house on the paper. The construction engineer can follow the design to build the actual building.

Constructing the actual functional system starting from a design specification is known as system implementation. Construction is invariably a project in itself and therefore must be planned beforehand. Planning and other implementation related issues are covered in this lesson.

7.2 WHAT IS IMPLEMENTATION?

After testing of the system, the candidate system is installed and implemented at the user's place. The old system is changed to a new or modified system and users are provided training to operate the new system. This is a crucial phase of SDLC and is known as implementation phase. Before discussing the activities of implementation phase, let us first see what is meant by implementation. The term 'Implementation' may be defined as follows:

Implementation is the process of converting the manual or old computerised system with the newly developed system and making it operational, without disturbing the functioning of the organisation.

7.3 TYPES OF IMPLEMENTATION

Implementation may be of the following three types:

(a) ***Fresh Implementation:*** Implementation of totally new computerised system by replacing manual system.

(b) ***Replacement Implementation:*** Implementation of new computerised system by replacing old computerised system.

(c) ***Modified Implementation:*** Implementation of modified computerised system by replacing old computerised system.

Software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g., a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements. Software maintenance reapplies each of the preceding phases to an existing program rather than a new one.

Four types of change are encountered during the maintenance phase. These four activities are undertaken after a program is released for use.

1. *Corrective Maintenance:* Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

2. ***Adaptive Maintenance:*** Over time, the original environment (e.g., CPU, operating system, business rules, external product characteristic) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate changes to its external environment.

3. ***Perfective Maintenance or Enhancement:*** As software is used, the user/customer will recognise additional functions that will provide benefit. Perfective maintenance extends the software beyond its original functional requirements.

4. ***Preventive Maintenance or Reengineering:*** Computer software deteriorates due to change, and because of this, preventive maintenance, often called software reengineering, must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

Only about 20 per cent of all maintenance work is spent 'fixing mistakes". The remaining 80 per cent is spent adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for future use. When maintenance is considered to encompass all of these activities, it is relatively easy to see why it absorbs so much effort.

7.4 PROCESS OF IMPLEMENTATION

Whatever be the kind of implementation, the implementation process has following two parts:

(i) Conversion

(ii) Training of Users

We will discuss these procedures in brief.

7.4.1 Conversion

Conversion is the process of changing from the old system to modified or new one. Many different activities are needed to be performed in conversion process depending upon the type of implementation (as defined above). During fresh implementation, all necessary hardware is installed and manual files are converted to computer-compatible files. During replacement implementation, old hardware may be replaced with new hardware and old file structures are also needed to be converted to new ones. The conversion process is comparatively simpler in the third type of implementation, i.e., modified implementation. In such implementation, existing hardware is generally not replaced and also no changes are made in file structures.

Conversion Plan

Before starting conversion process, the analyst must prepare a plan for conversion. This plan should be prepared in consultation with users. The conversion plan contains following important tasks:

(i) Selection of conversion method;

(ii) Preparation of a conversion schedule;

(iii) Identification of all data files needed to be converted;

(iv) Identification of documents required for conversion process;

(v) Selecting team members and assigning them different responsibilities.

Conversion Methods

The following four methods are available for conversion process:

(a) ***Direct Cutover:*** In this method, the old system (whether manual or computerised) is completely dropped out on one particular date and new system is implemented.

(b) ***Parallel Conversion:*** In this method, the old method is not dropped out at once, but both old and new systems are operated in parallel. When new system is accepted and successfully implemented, old system is dropped out. The organisation can still fall back to the old system without loss of time and money.

The disadvantages of the parallel system approach are:

- It doubles operating costs.
- The new system may not get fair trial.

(c) ***Phase-in-method of Conversion:*** In this method, the new system is implemented in many phases. Each phase is carried out only after successful implementation of previous phase. The old system is used until a planned conversion day, when it is replaced by the new system. These are no parallel by the new system. There are no parallel activities. The organisation relies fully on the new system. The main disadvantage of this approach are; no other system to fall back on, if difficulties arise with new system. Secondly, wise and careful planning is required.

(d) ***Pilot System:*** In this method, only a working version of new system is implemented in one department of the organisation. If the system is accepted in that department, it is implemented in other departments the either in phases or completely. This method is used when it is not possible to install a new system throughout an organisation all at once. The conversion of files, training of personnel or arrival of equipment may force the staging of the implementation over a period of time, ranging from weeks to months. It allows some users to take advantage of the new system early. Also it allows training and installation without unnecessary use of resources.

Each of the above methods has its advantages and disadvantages. Although, the direct cutover is the fastest way of implementing the system, this method is very risky. As the organisation will completely depend on the new system, if it fails, there would be a great loss to the company. Parallel method is considered more secure, but it has many disadvantages. The parallel conversion doubles not only the operating costs but also the workload. The major disadvantage of parallel conversion is the outputs of both the systems may mis-match and in such cases, it becomes very difficult for management to analyse, compare and evaluate their results. Although, phase-in-method and pilot systems are more time-consuming methods of implementation, they are considered more reliable, secure and economical.

7.4.2 Training of Users

Training of users is another major part of implementation. It is considered most important part of software development, as it helps users in operating and maintaining the system. There are many methods and aids used by the systems analyst to provide training to users. The different methods and aids of training are:

(a) ***Training Programmes and Seminars:*** Many programmes and seminars are conducted by the analyst at the user's place for providing them the required training for converting and operating the new system.

(b) ***User's Manual:*** User's manual is the training guide that contains complete description on how to implement and operate the candidate system. It has to be prepared with development of every new or modified system. It must contain the following:

(i) Hardware requirements specifications

(ii) Software requirements specifications

(iii) Installation procedure

(iv) Input screen and output formats

(v) Menu and sub-menus

(vi) Command lists

(vii) Procedures to execute each option of the menu

(viii) Data file layouts

(ix) Conversion procedures

(x) Guidelines to operate the system

(xi) List of errors with explanations and debugging methods.

(c) ***Help Screens:*** Helps screens are the useful aids to provide training to the users in operating the system. They provide online help to the users during execution of the system. These screens must be designed by the analyst or other training personnel during implementation phase of SDLC.

(d) ***In-house Training:*** The main advantage of offering in-house training is that instruction can be tailored according to the requirements of the organisation. Often the vendors negotiate fees and charges that are more economical so that company can involve more personnel in the training program than is possible when travel is required. However, the disadvantage of distracting telephone cells, business emergencies and other interruptions must not be overlooked.

Although high-quality training is an essential step in systems implementation, yet it is not sufficient by itself.

7.5 MAINTENANCE

Implementation is not the last phase of SDLC. As the user requirements may change in future, it becomes essential for the developer to maintain the system. After completion of implementation phase, the software is required to be properly maintained. If the system s not properly maintained, it may fail too. Therefore, generally more than 50 per cent of total software development time is spent in maintaining the system.

Maintenance is actually improvement and updation of software in response to the changes in the organisation. It includes many activities including following:

- Correcting design errors
- Correcting coding errors

- Updating documentation and test data
- Adding, modifying or redeveloping the code
- Regular acceptance and validation testing.

Maintenance is quite labour-intensive and expensive phase of a software development project. A system may become difficult to be maintained if it is not properly designed. An object-oriented system is considered easier to be maintained due to its feature of reusability of the code. Maintenance costs can be reduced by making a maintenance reduction plan. This plan includes maintenance management and software modification audits. Maintenance Management Audits evaluate the quality of the maintenance effort, while Software Modification Audits evaluate and update the programs.

7.6 REVIEW PLAN

After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against pre-defined requirements. It determines how well the system continues to meet performance specification. It also provides information to determine whether major re-design or modification is required.

A post-implementation review is an evaluation of a system in terms of the extent to which the system accomplishes stated objectives and actual project costs exceed initial estimates. It is usually a review of major problems that need converting and those that surfaced during the implementation phase.

The post implementation study begins with the review team, which gathers and reviews requests for evaluation. Unexpected change in the system that affects the user or system performance is a primary factor that prompts system review. Once request is filed, the user is asked how well, the system is functioning to specifications or how well the measured benefits have been realised, suggestions regarding changes and improvements are also asked for.

7.7 HARDWARE/SOFTWARE SELECTION

Implementation of a computerized system will invariably involve hardware and software. In some cases the existing hardware/software may need updating while in some other cases procuring a whole new set of system may be need.

The system analyst has to determine, what software package is best for the candidate system and, where software is not an issue, the kind of hardware and peripherals needed for the final conversion. To do the job well, the analyst must be familiar with the computer industry in general, what various computers can do or cannot do, whether to purchase or lease a system, the vendors and their outlets, and the selection procedure.

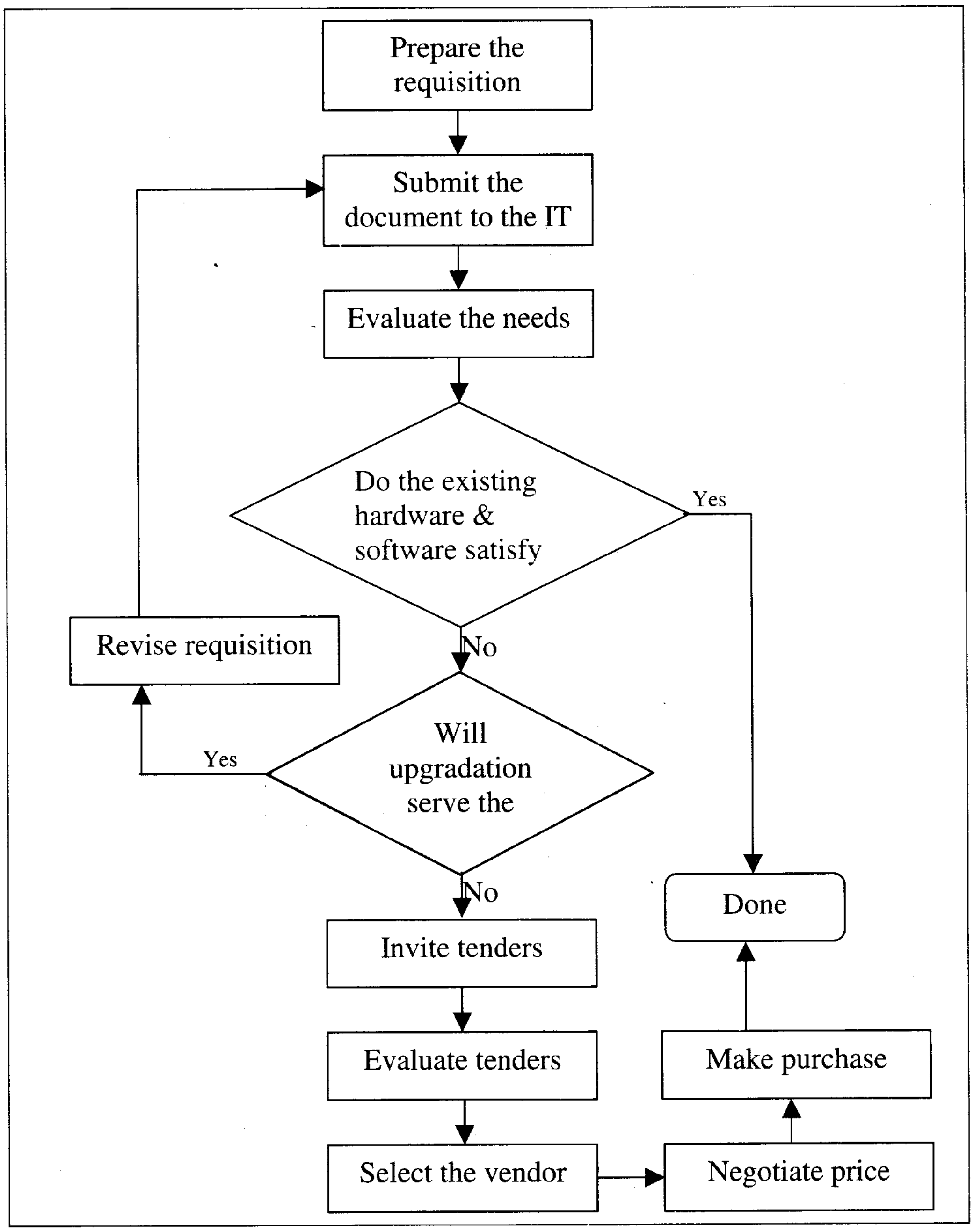


Figure 7.1

7.7.1 Procedure for Selection: Computer Contract

The selection of hardware and software begins with requirement analysis followed by a request for proposal and vendor evaluation. The final system selection initiates contract negotiation. It includes purchase price, maintenance agreements, and the amount of updating or enhancement to be available by the vendor over the life of the system. And the next step is hardware acquisition.

The procedure for hardware/software selection will vary from one implementation to another. However, some guideline needs to be in place that may be followed while making selection. A generic procedure is presented below which can be tailored to suit the occasion.

Please note that this is just one possible selection procedure highlighting the importance of having a well defined and systematic procurement policy in the organization. The criteria that must be taken into consideration while making a selection for the software and hardware are discussed below.

Major Phases in Selection

The selection procedure may be divided into a number of phases as listed below:

1. ***Phase-1:*** Selecting an optimal vendor
2. ***Phase-2:*** Request from proposal (RPF) document preparation
3. ***Phase-3:*** RPF review and vendor recommendation
4. ***Phase-4:*** RPF analysis and vendor selection
5. ***Phase-5:*** Contract negotiation
6. ***Phase-6:*** Summary report

7.7.2 Make vs. Buy Decision

Buying a product from a new business, such as the explosive software industry poses unusual problems. Customers cannot evaluate decades of performance history by the company, and not enjoying the benefits of an objective "consumer report" on new products, they often feel at the mercy of fast-talking salespeople. Therefore, it is important for people in the market for software to ask some really tough but relevant questions. Any reputable supplier should be able to answer the following 15 questions without backpedaling.

(i) ***Range of Products:*** Can you offer us a complete range of software system designed to work together? Or will we have to piece together a patchwork of systems to fully computerize our organization?

(ii) ***Decision Support Systems:*** Are your systems just record keepers, or can they really help us make decisions? Can we pull together information from any of our integrated systems in the desired form.

(iii) ***In-house Development:*** Can you provide business software for both mainframe and microcomputers? Do you develop this software yourself or do you simply market it for another company?

(iv) ***Online:*** Are your systems truly online? How many of your systems are online? How secure are they?

(v) ***Debugging and Testing:*** Will my company have to be the one that discovers the bugs in your brand new system? Just how long have your systems actually been used, and how have they been tested?

(vi) ***Updates:*** Will you update your systems as technology advances and regulations change? What are some of your most recent updates? Will you keep us current on regulatory change?

(vii) ***Flexibility/Adaptability:*** Are your systems really adaptable to our unique needs? Or will we have to change or add to them ourselves to get the features we want?

(viii) ***History/Performance:*** How long have you been in business? What are your revenues? What is your growth record? Where will your company be in five years from now? Can you show me an annual report?

(ix) ***Other Customers:*** How many systems has your company installed? How many of these were installed in the past six months? How many of your earlier customers are still using and liking your systems?

(x) ***Security:*** Are your systems secure? Do you provide password type protection and to how many levels? What other type of security provisions do your systems have?

(xi) ***Networking:*** Can you link our executives' personal computers directly to the mainframe, so they can get their own information? Is that software available right now?

(xii) ***Training Support:*** How will you make sure our own people thoroughly understand your system? Do you have educational centers near us or will we have to travel all the way across the country to find one? Will you be there to help during installation and after?

(xiii) ***In-House Specialists:*** How many of your people specialize in software for my industry? How many accountants work for you? Human resource specialists? Manufacturing experts?

(xiv) ***Special Features:*** Do your systems have built-in features that make them easier to use? What happens if someone needs help figuring out a feature? Do you have online documentation that is easy to understand?

(xv) ***Upgrading:*** As my business changes will your system be flexible enough to change with it? Or will we have to pay a lot to revamp it? Or even regenerate it?

7.7.3 Criteria for Software Selection

Software selection is a critical aspect of system development. As mentioned earlier, the search starts with the software, followed by the hardware. There are two ways of acquiring software: custom-made or "off-the-shelf" packages. Today's trend is towards purchasing packages, which represent roughly 10 per cent of, what it costs to develop the same in-house. In addition to reduced cost, there are other advantages:

1. A good package can get the system running in a matter of days rather than weeks or months required for "home-grown" packages
2. MIS personnel are released for other projects
3. Packages are generally reliable and perform according to stated documentation
4. Minimum risks are usually associated with large-scale systems and programming efforts
5. Delays in completing software projects in-house often occur because, programmers quit in midstream
6. It is difficult to predict the cost of "home-grown" software
7. The user has a chance of seeing, how well the package performs before purchasing it

There are drawbacks, however, to software packages:

1. The package may not meet user requirements adequately.
2. Extensive modification of a package usually results in the loss of vendor support.
3. The methodology for package evaluation and selection is often poorly defined. The result is a haphazard review based on a faulty process or questionable selection criteria.
4. For first-time software package users, the overall expectation from package is often unclear and ill-defined.

It can be seen, then, that the quality of a software package cannot be determined by price alone. A systematic review is crucial. Prior to selecting the software, the project team must set up criteria for selection. Selection criteria fall into the categories described here.

Reliability

It is probable that the software will execute for a specified time period without a failure, weighted by the cost to the user of each failure encountered. It relates to the ease of recovery and the ability to give consistent results. Reliability is particularly important to the professional users.

For example, a pharmacist relies on past files on patients, when filling prescriptions. Information accuracy is crucial.

Hardware may become inoperative-because of design errors, manufacturing errors, or deterioration caused by heat, humidity, friction, and the like. In contrast, software does not fail or wear out. Any reliability problems are attributable to errors introduced during the production process. Furthermore, whereas, hardware failure is based lately on random failures, software reliability is based on predestined errors.

Although, reliable software is a desirable goal, limited progress has been made towards improving it in the last decade. The fact of unreliable software had led to the practice of securing maintenance agreements after the package is in operation. In a sense, unreliability is rewarded.

Software reliability brings up the concept of modularity, or the ease with which a package can be modified. This depends on, whether the package was originally designed as a package or was retrofitted after its original development for single installation use. A package with a high degree of modularity has the capacity to operate in many machine configurations and perhaps, across manufacturers' product lines.

With modularity comes expandability, which emphasizes the sensitivity of a software package to handle an increased volume of transactions or to integrate with other programs. The following questions should be considered:

1. Is there room for expanding the master file?
2. How easily can additional fields, records, and files be added?
3. How much of the system becomes unusable when a part of it fails?
4. Are there errors a user can make that will bring down the system?
5. What are the recovery capabilities?

Functionality

It is a definition of the facilities, performance, and other factors that the user requires in the finished product. All such information comes from the user. The following are key questions to consider:

1. Do the input transactions, files and reports contain the necessary data elements?
2. Are all the necessary computations and processing performed according to specifications?

Capacity

Capacity refers to the capability of the software package to handle the user's requirements for size of files, number of data elements, volume of transactions and reports, and number of occurrences of data elements. All limitations should be checked.

Flexibility

It is a measure of the effort required to modify an operational program. One feature of flexibility is adaptability, which is a measure of the ease of extending the product.

Usability

This criterion refers to the effort required to operate, prepare the input and interpret the output of a program. Additional points to be considered are portability and understandability. Portability refers to the ability of the software to be used on different hardware and operating systems. Understandability means that, the purpose of the product is clear to the evaluator and that, the package is clearly and simply written is free of jargons, and contains sufficient references to readily available documents so that the reader can comprehend advanced contents.

Security

It is a measure of the likelihood that, a system's user can accidentally or intentionally access or destroy unauthorized data. A key question is: How well can one control access of software or data files? Control provides system integrity.

Performance

It is a measure of the capacity of the software package to do what it is expected to do. This criterion focuses on throughput, or how effectively a package performs under peak loads. Each package should be evaluated for acceptance on the user's system, the language in which a package is written and the operating system and additional performance considerations. If we plan to modify or extend a package, it is easier if it is written in a language that is commonly known to programmers. Likewise, if the package runs only under a disk operating system and the installation is under a full operating system, then, either the package will have to be upgraded to the larger operating system or the system downgraded to handle the package as it is. In either case, the change could be costly and counterproductive.

Serviceability

This criterion focuses on documentation and vendor support. Complete documentation is critical for software enhancement. It includes a narrative description of the system, system logic and logic flow charts, input-output and file descriptions and layouts, and operator instructions. Vendor support assures the user adequate technical support for software installation, enhancements, and maintenance.

The user should determine, how much on-site technical assistance the vendor provides, especially, during the first few weeks after the installation.

The user expects on-site training and support as part of most commercial packages. It is vital to inquire about the amount of training provided. The user may require training at several levels-clerical, operations, programming and management.

Ownership

Who owns the software once it is "sold" to the user? Most of the standard license agreement forms, essentially, lease the software to the user for an indefinite time. The user does not "own" it, which means that, the source code is inaccessible for modification, except by the vendor. Many users enter into an escrow arrangement, whereby, the vendor deposits the source code with a third-party escrow agent, who agrees to release the code to the user, if the vendor goes out of business or is unable to perform the services specified in the license.

In acquiring software, several questions should be asked:

1. What rights to the software is the user buying?
2. Can the user sell or modify the software?
3. If the vendor is modifying the package, especially for the user, can the vendor sell it to others within the same industry the user is in?
4. What restrictions are there to copying the software or documentation?"

Minimal Cost

Cost is a major consideration in deciding between in-house and vendor software. Cost-conscious users consider the following points:

1. Development and conversion costs.
2. Delivery schedule.
3. Cost and frequency of software modifications.
4. Usable life span of the package.

The table given below summarizes these criteria:

Criterion	**Meaning**
1. Reliability	Gives consistent results
2. Functionality	Functions to standards
3. Capacity	Satisfies volume requirements
4. Flexibility	Adapts to changing needs
5. Usability	Easy to operate and understand-usually-friendly
6. Security	Maintains integrity and prevents unauthorized access

Contd...

7. Performance	Capacity to deliver as expected
8. Serviceability	Good documentation and vendor support
9. Ownership	Right to modify and share the use of package
10. Minimal costs	Affordable for intended application.

Check Your Progress

1. Fill in the blanks:
   (a) .... is the process of changing from the old system to modified or new one.
   (b) .... is the training guide that helps the user to operate the candidate system.
   (c) Implementation of new computrised system by replacing old computerised system is called .... implementation.
   (d) Training of users is a part of .... process.
2. State true or false:
   (a) The conversion process is simpler in replacement implementation than modified implementation
   (b) Maintenance is not the phase of SDLC.
   (c) Software Modifications Audits evaluate and update the programs during maintenance.

7.8 LET US SUM UP

The candidate system is installed after testing of the system and implemented at the user's place. Implementation is the process of converting the manual or old computerized system with the newly developed or old computerized system with the newly developed system and making it operational, without disturbing the functioning of the organization. The three types of implementation include fresh implementation, replacement implementation, modified implementation. The four activities undertaken after the release of a program include corrective maintenance, adaptive maintenance, perfective maintenance and preventive maintenance.

The implementation process has two parts: conversion and training of users. The methods available for conversion are direct cutover, parallel conversion, phase-in-method and pilot system. The different aids of training are training programmes, seminars and user's manuals. The conversion plan should anticipate possible problem and methods for controlling them. After that, a review should be conducted to determine whether the system is meeting expectations and where improvement are needed.

After complementation of implementation phase, the software is required to be properly maintained. A System may become difficult to be maintained if it is not properly designed.

7.9 KEYWORDS

Implementation: The process of converting the manual or old computerized system with the newly developed system and making it operational, without disturbing the functioning of the organization.

Fresh Implementation: Implementation of totally new computerized system by replacing manual system.

Replacement Implementation: Implementation of new computerized system by replacing old computerized system.

Modified Implementation: Implementation of modified computerized system by replacing old computerized system.

Conversion: The process of changing from the old system to modified or new system.

Direct Cutover: The process of dropping out one old system on a paricular and implementation of new system.

Parallel Conversion: The process where both old and new systems are operated in parallel.

User's Manual: Training guide that contains complete description on how to implement and operate the candidate system.

Post-implementation Review: Evaluation of a system in terms of the extent to which the system accomplishes stated objectives and actual project costs exceed initial estimates.

Maintenance: Improvement and updation of software in response to the changes in the organization.

7.10 QUESTIONS FOR DISCUSSION

1. What is implementation? List various types of implementation.
2. List various changes encountered during the maintenance phase.
3. What are the various processes of implementation? Describe them.
4. What is the function of conversion plan?
5. Write a short note on review plan.
6. What is maintenance? List various activities included in maintenance.
7. Write short notes on the following:
   (a) Training of Users
   (b) conversion
8. How important is the reliability factor in software packages? Discuss.
9. It has been suggested that software should be considered before hardware. Do you agree? Why?
10. What software criteria are considered for selection? Summarize.

Check Your Progress: Model Answers

1. (a) Implementation

   (b) User's Manual

   (c) Direct cutover

   (d) Implementation

2. (a) False

   (b) True

   (c) False

7.11 SUGGESTED READINGS

James C Wetherbe, St. Paul, *Systems analysis and design,* West Pub. Co., ©1988.

A Ziya Aktas, Englewood Cliffs, N.J., *Structured analysis and design of information systems*, Prentice-Hall, ©1987.

Pankoj Jalote, *An Integrated Approach to Software Engineering*, Springer.

LESSON 8

PROJECT SCHEDULING

CONTENTS

8.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Describe various reasons for the initiation of a project
- Describe various sources of projects' requests
- Understand how to manage project review and selection
- Describe the role of analyst as a preliminary investigator
- Define a problem and evaluate it

8.1 INTRODUCTION

System analysts do not start working on any projects they desire. They receive a lot of request from the management for starting different type of the projects. When projects are formally requested, the systems analysts, under the managements direction, conduct a preliminary investigation to analyse the reasons for the request and collect various facts to respond to the request in a systematic way. Some projects are feasible, while others may not be feasible for various reasons.

Developing a new information system is one kind of a planned organizational change. The introduction of a new system includes purchase of new hardware and software, appointment of skilled persons and/or training of existing staff for using the new system and changes in management and organization. So, developing a new system requires redesigning the organization. As an organization always has many projects (concerned with MIS or other business functions) to do, the management must decide the priority of a system development in order to streamline the business.

8.2 REASONS FOR DEVELOPING NEW SYSTEMS PROJECT

System projects are initiated for different reasons. The most important reasons are:

1. ***Capability:*** Business activities are influenced by an organization's ability to process transactions quickly and efficiently. Information systems add capability in three ways:
   (i) *Improved Processing Speed:* The inherent speed with which computers process data is one reason why organisation seek the development of systems projects.
   (ii) *Increased Volume:* Provide capacity to process a greater amount of data, perhaps to take advantage of new business opportunities.
   (iii) *Faster Retrieval of Information:* Locating and retrieving information from storage. The ability in conducting complex searches.
2. ***Control:***
   (i) *Greater Accuracy and Consistency:* Carrying out computing steps, including arithmetic, correctly and consistently.
   (ii) *Better Security:* Safeguarding sensitive and important data in a form that is accessible only to authorised personnel.
3. ***Communication:***
   (i) *Enhanced Communication:* Speeding the flow of information and messages between remote locations as well as within offices. This includes the transmission of documents within offices.
   (ii) *Integration of Business Areas:* Coordinating business activities taking place in separate areas of an organisation, through capture and distribution of information.
4. ***Cost:***
   (i) *Monitor Costs:* Tracking the costs of labour, goods and overhead is essential to determine whether a firm is performing a line with expectations-within budget.
   (ii) *Reduce Costs:* Using computing capability to process data at a lower cost than possible with other methods, while maintaining accuracy and performance levels.

5. ***Competitiveness:***
   (i) *Lock in Customers:* Changing the relationship with and services provided to customers in such a way that they will not think of changing suppliers.
   (ii) *Lock Out Competitors:* Reducing the chances of entering the competitors in the same market because of good information systems being used in the organisation.
   (iii) *Improve Arrangements with Suppliers:* Changing the pricing, service or delivery arrangements, or relationship between suppliers and the organisation to benefit the firm.
   (iv) *New Product Development:* Introducing new products with characteristics that use or are influenced by information technology.

8.3 SOURCES OF PROJECTS REQUESTS

There are mainly four primary sources of project requests. The requesters inside the organization are:

- Department Managers
- Senior Executives
- System Analysts

In addition, government agencies outside the organisation may also ask for information systems projects.

8.3.1 Requests from Department Managers

Frequently, department managers who deal with day-to-day business activities, are looking for assistance with their departments. They are often not satisfied with the amount of ime that the staff takes to complete the job. Sometimes, they feel that the staff members are involved in duplication of work also. In this case, the manager will discuss this problem with other administrators regarding their clerical as well as processing work and persuade higher authority to approve the development of a computer based system for office administration.

8.3.2 Requests from Senior Executives

Seniors executives like Presidents, Vice-Presidents usually have more information about the organization as compared to department managers. Since these executives manage the entire organization, so naturally they have broader responsibilities. Obviously, systems project requests submitted by them carry more weightage and are generally broader in scope also.

8.3.3 Requests from System Analysts

Sometimes systems analysts finalize areas where it is possible to develop projects. In such cases, they may prefer either writing systems proposal themselves or encouraging a manager to allow the writing of a proposal on their behalf. For instance, in an organisation, an analyst sees that the library information system takes more time in processing and is inefficient, may prepare a project proposal for a new library information system. But the direction of the analyst, who is fully aware about the new technology that improves the existing library information system, the librarian may initiate the development of information system to the higher authority for approval.

8.3.4 Requests from Outside Groups

Developments outside the organisation also lead to project requests. For example, government contractors are required to use special cost accounting systems with government stipulated features. Generally, it has been observed that new demands from external groups bring about project requests, either for new systems or changes in current ones. Project requests originated from this source are also quite important.

8.4 MANAGING PROJECT REVIEW AND SELECTION

It is true that a number of requests for systems development are generated in the organisation. Someone in the organisation must decide which requests to pursue and which to reject.

The management decides the priority of a system development by reviewing the answers of the following questions:

- When should the organization go for computerization, if it is still using manual systems?
- Are the users satisfied with the performance of existing systems (manual/computerized)? If not, what are the reasons?
- What are the major problems of the existing systems? Do they affect the normal working of the organization? If yes, how long can the organization tolerate such problems?
- What are the major projects that the organization has to do? Which one of all these is the most important?

The management must ensure that the most important systems are developed first, followed by the less important ones and the least important ones in the last. The criteria to accept or reject a request can be decided in a number of ways. One of the suitable methods commonly in use is by committee. Mainly three committees formats are commonly used:

1. ***Steering Committee:*** This is one of the most common methods of reviewing and selecting projects for development. Such a committee, consisting of key managers from various departments of the organisation as well as members of information systems group, is responsible for supervising the review of project proposals. This committee receives requests for proposal and evaluates them. The main responsibility of the committee is to take decision, which often requires more information than the proposal provides. It is, therefore, desired to have preliminary investigation to gather more details. The steering committee approach is generally favoured because systems projects are considered as business investments. Management, not systems analysts or designers, selects projects for development. Decisions are made on the basis of the cost of the project, its benefits to the organisation and the feasibility of accomplishing the development within the limits of information systems technology.
2. ***Information Systems Committee:*** In some organization, the responsibility for reviewing project requests is entrusted to a committee of managers and analysts in the information systems department. Under this method, all requests for service and development are submitted directly to a review committee within the information systems department. This committee approves or disapproves projects and sets priorities, indicating which projects are most important and should receive immediate attention. This method can be used when many requests are foı routine services or maintenance of existing applications. When major equipment decisions are required or

when long-term development commitments are needed to undertake a project, the decision authority decided whether a project should proceed or not. So, the major functions of this committee are:

(i) To review the systems plan and approve/disapprove them.

(ii) To integrate the systems that share the input data.

(iii) To provide alternatives to the project.

3. ***User Group Committee:*** In some organization, the responsibility for project decisions is entrusted to the users themselves. Individual departments hire their own analysts and designers who handle project selection and carry out development. Although the practice of having user committees for both choose and develop systems does take some of the burden from the systems development group it can have disadvantages for the users. Some users' groups may find themselves with defective or poorly designed systems that require additional time and effort to undo any damage caused by the misinformation that such systems could generate. Although users groups may find the decisions of steering committees and Information Systems Committees disappointing at times, the success rate for users who undertake development job is not very encouraging.

8.5 PROJECT REQUEST CONTENTS

The project proposals submitted by the users or the analysts to the Project Selection Committee is a critical element in launching the systems study. There is a general agreement that a project request form should contain the following:

- What is the problem?
- What are the details of the problem?
- How significant is the problem?
- What does user feel is the solution?
- How will the information systems help?
- Who else knows about this and could be contacted?

The project selection committee is responsible to review the proposals carefully and finally selects those projects which are most beneficial to the organization. Therefore, a preliminary investigation is often requested to gather details which are asked in the project request-forms.

8.6 PRELIMINARY INVESTIGATION

The first step in the system development life cycle is the preliminary investigation to determine the feasibility of the system. The purpose of the preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the business system in all respect. Rather, it is the collecting of information that helps committee members to evaluate the merits of the project request and make an informed judgement about the feasibility of the proposed project.

Analysts working on the preliminary investigation should accomplish the following objectives:

1. Clarify and Understand the Project Request. What is being done? What is required? And why? Is there an underlying reason different from the one the user identifies?
2. Determine the size of the project.
3. Access costs and benefits of alternative approaches.
4. Determine the technical and operational feasibility of alternative approaches.
5. Report the finding to management, with recommendations outlining the acceptance or rejection of the proposal.

Conducting the Investigation

The data collected by the analysts during preliminary investigations are gathered through three primary methods:

1. ***Reviewing Organisation Documents:*** The analysts conducting the investigation first learn about the organisation involved in, or affected by the project. For example, to review an inventory systems proposal means knowing first how the department works and who are the persons directly associated with inventory system. Analysts can get some details by examining organisation charts and studying written operating procedures. The procedures clearly define various important steps involved in receiving, managing and dispersing stock.
2. ***On-Site Observations:*** In this method, the analysts observe the activities of the system directly. One purpose of on-site observation is to get as close as possible to the real system being studied. During on-site observation, the analyst can see the office environment, work local of the system and the users, methods of work and the facilities provided by the organisation to the users.
3. ***Conducting Interviews:*** The above two methods tell the analysts how the system should operate, but they may not include enough details to allow a decision to be made about the merits of a system proposal, nor do they present user views about current operations. Analysts use interview to learn these details. Interviews allow analysts to learn more about the nature of the project request and the reason for submitting it. Interview should provide details that further explain the project and show whether assistance is merited economically, operationally and technically.

8.7 PROBLEM CLASSIFICATIONS AND DEFINITIONS

One of the most difficult tasks of system analysts is developing a clear, in-depth understanding of the problem being investigated, without which it becomes impossible to specify the requirements for a new project with any accuracy. Several questions should be posed for this, as:

(i) What is the problem?

(ii) How complex is it?

(iii) What are its likely causes?

(iv) Why is it important that the problem be solved?

(v) What are possible solutions to the problem?

(vi) What types of benefits can be expected once the problem is solved?

8.7.1 Defining a Problem

It takes considerable skill to determine the true cause of a systems problem. A system analysts determines if the problem can be classified according to one or more common types of systems problems. With a knowledge of the common types of problems, the analyst can diagnose a problem by examining its characteristics. The following examples illustrates these findings.

(a) ***The Problem of Reliability:*** A system suffers from the problem of reliability when procedures work some but not all of the time, or when use of the same procedure leads to different results. Analysts must work continually to improve the reliability of systems by running software tests to document that too runs of a computer program lead to identical results, by selecting equipment with low failure rates, and by monitoring processing schedules to ensure that results are on time.

(b) ***The Problem of Validity:*** Maintaining validity in computer software is a troublesome design problem. The objective in design is to produce a flawless product, one that will always reflect actual events. Validity problems result when the environment changes and these changes are not incorporated into the software.

(c) ***The Problem of Accuracy:*** A system is inaccurate when processing is error-prone. 'Routine' transaction-based manual procedures are basically suitable for conversion to computer-based methods of processing because the computer is far more accurate than human beings, provided that software is written properly.

(d) ***The Problem of Economy:*** Project with clear-cut savings are likely to be considered suitable for conversion to computer-based methods of processing. Much like the problem of accuracy, the problem of economy is relatively easy to identify, the danger with the problem of economy is the naïve assumption by both users and system managers that the computer will eliminate the cause of the problem. Budget managers will say that this assumption is not always true; they will report that some project cost far more than they return. Thus, before moving ahead, on a project assignment, the analyst must ask, "Is the project work doing".

A partial answer to this question follows from determining the return on the investment expected from the project. If the return is low, more economical projects should be selected.

(e) ***The Problem of Timeliness:*** The problem of timeliness relates more to the transmission of information than to the processing or storing of it. If information is available but cannot be retrieved when and where it is needed, the system suffers from the problem of timeliness. Organizations have committed extensive resources to handle the problem of timeliness in recent years. Fingertip access to information has been the desired objective. Only when retrieval problems are small and well defined has the overall success rate improved.

(f) ***The Problem of Capacity:*** The problem of capacity occurs when a system component is not large enough. This problem is specially common in organizations that experience peak periods of business or that are rapidly growing. With growth, smaller-capacity equipment soon becomes too small; smaller staff groups soon become overworked. In either case, some expansion is needed to handle the increasing volume of business.

(g) ***The Problem of Throughput:*** The problem of throughput may be viewed as the reverse of the problem of capacity. Throughput deals with the efficiency of a system. If system capacity is high and production low, a problem of throughput occurs. Similar to the problem of capacity, the problem of throughput may be much easier to spot than to treat. Rather, a manager must be able to determine the root of the problem for any improvement in throughput.

8.7.2 Evaluating the Problem

Suppose that a problem has been identified. The next step is problem evaluation. It consists of asking the following questions:

- Why is it important to solve the problem?
- What are possible solutions to the problem?
- What types of benefits can be expected once the problem is solved?

8.7.3 Sources of Problem/Opportunity

Organisations usually face problems or have opportunity due to the following:

- A new product or plant or branch
- A new market or new process
- Failure of an existing system
- Inefficiency of an existing system
- Structural errors in the existing system, etc.

Thus, a thorough analysis of the situation need to be required. Not only the above listed reasons but there exist some organisation-based reasons too.

8.7.4 Problem Identification and Definition

For identifying problems/opportunities, we scan the following:

- The performance of the system
- The information being supplied and its form
- The economy of processing
- The control of the information processing
- The efficiency of the existing system
- The security of the data and software
- The security of the equipment and personnel , etc.

After identification of the problem, it is defined and a general direction or method for solving this problem is also determined. Then project boundaries are defined. The management established the term of reference as well as the resources to be provided for the project. System development is an iterative process and the first identifiable stage of it is Problem Definition whose final output is Terms of Reference.

Check Your Progress

1. Fill in the blanks:
   (a) Business activities are influenced by an organization's ability to .... quickly and efficiently.
   (b) The primary sources of project requests include ...., ..., ... and ....

Contd....

(c) The .... decides the priority of a system development.

(d) The main responsibility of the .... committee is to take decision, which often requires more information than the proposal provides.

(e) The first step in the system development life cycle is the ....

2. State True or False

(a) The analyst studies the problems and needs of an organization during feasibility and requirements analysis phases of SDLC.

(b) Senior executives usually have more information about the organization as compared to department managers.

(c) The direction of the analyst need not help in the improvement of the existing system.

(d) Interviews are conducted for preliminary investigation.

8.8 LET US SUM UP

Developing a new information system is one kind of a planned organizational change. System projects are initiated to enhance the capability, control, communication and competitiveness and also to monitor and reduce cost of the system. Department managers, senior executives and system analysts are the main sources to generate a system's request. In addition government agencies may also ask for Information Systems projects.

Our of a number of requests for systems development the management decides the priority of a system development. The three committees for reviewing and selecting projects for development include steering committee, information system committee and user group committee.

The first step in SDLC, preliminary investigation, determine the feasibility of the system. The data collected by the analyst are gathered through reviewing organization documents, on-site observations and conducting interviews. After that the problem is defined classified, and evaluated on the bases of various characteristics.

8.9 KEYWORDS

Steering Committee: A committee consisting of key managers from various departments of the organization as well as members of information systems group.

Information Systems Committee: A committee of managers and analysts in the information systems department.

8.10 QUESTIONS FOR DISCUSSION

1. List some most important reasons for a project to be initiated.
2. What are the primary sources of project request?
3. List the questions reviewed by the management to decide the priority of a system development.
4. What are the functions of a long-term development committee?

5. List some of the contents of a project request.
6. What is preliminary investigation? Why is it done?
7. What are the objectives of preliminary investigation?
8. What is a problem? What are the various types of problem? How can a problem evaluated?
9. What are the various sources of a problem? How can you identify a problem?
10. Why do we develop a new systems project?
11. Describe different committees formats used for managing projects.
12. From where can an analyst collect data during preliminary investigations?

Check Your Progress: Model Answer

1. (a) Process transactions
   (b) Department managers, Senior executives, System analysts, government agencies
   (c) Management
   (d) Steering
   (e) Preliminary investigation
2. (a) True
   (b) True
   (c) False
   (d) True

8.11 SUGGESTED READINGS

Erik W. Larson, Clifford F. Gray, *Project Management*, McGraw-Hill Professional.

S.A. Kelkar, *Structured System Analysis and Design*, Prentice Hall of India.

Pankoj Jalote, *An Integrated Approach to Software Engineering*, Springer.

UNIT V

LESSON

9

SYSTEM TESTING AND QUALITY ASSURANCE

CONTENTS

9.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Define system testing, reliability and quality assurance
- Describe system security and its needs

- Describe various forms of computer crimes
- Describe various system control
- Define system audit
- Describe security measures on a network
- Describe disaster recovery and computer ethics

9.1 INTRODUCTION

After the development and testing of the system, the major stress is given on system control, its reliability and security i.e., to assure the quality of the system. Quality assurance is an integral part of all production and development processes. Reliability is the degree to which the system performs its intended functions over a span of time. System Security is required to prevent the system from unauthorized access or any natural disaster such as fire, flood earthquake etc. With the advent in the field of technology, Cyber crimes are developing day by day to destroy the system or to malfunction it. Systems should be protected from these cyber criminals, so specific security measures should be designed to prevent the system from trapping, unauthorized access and virus attacks. Besides security aspects, adequate control measures are needed in the system due to its reliability and correctness. Some recovery measures are also to be planned to recover the system after any natural disaster. Encryption techniques can be used to avoid unauthorized access of the data.

9.2 SYSTEM TESTING

'System Testing' concentrates on testing the system as a whole.

In a classic venture, 'unit testing' is performed by the programmers. This makes sure that the individual components are functioning OK. The 'Integration testing' concentrates on victorious incorporation of all the individual portions of software.

After the integration of components, the system as a whole is required to be thoroughly tested to make sure that it fulfills the Quality principles.

So the System testing builds on the earlier levels of testing such as unit testing and Integration Testing.

Typically a dedicated testing team is accountable for doing 'System Testing'. System Testing is a vital step in Quality Management Process.

- In the Software Development Life cycle System Testing is the first level where the System is tested as a whole.
- The System is tested to confirm if it fulfills the functional and technical requirements.
- The application/System is tested in an environment that closely resembles the production environment where the application will be finally deployed.
- The System Testing enables us to test, verify and validate both the Business requirements along with the Application Architecture.

9.3 WHAT IS SYSTEM CONTROL AND RELIABILITY?

Quality assurance is an integral part of all production and development processes. It ensures the quality of a product by providing adequate measures that are built into the system in order to ensure the quality of the developed software. System control governs all the activities of the software development process. Reliability is the degree to which the system performs its intended functions over a span of time. Reliability is the basic goal of system control.

9.4 SYSTEM RELIABILITY

There is no doubt that the reliability of a computer system is an important element of its overall quality. If a system repeatedly and frequently fails to perform, it matters little whether other software quality factors are acceptable.

System reliability, unlike many other quality factors, can be measured, directed, and estimated using historical and developmental data. System reliability is defined in statistical terms as "the probability of failure free operation of a computer system in a specified environment for a specified time."

Failures can be only annoying or catastrophic. One failure can be corrected within seconds while another requires weeks or even months to correct. Complicating the issue even further, the correction of one failure may in fact result in the introduction of other errors that ultimately result in other failures.

9.5 MEASURES OF RELIABILITY AND AVAILABILITY

Early work in system reliability attempted to extrapolate the mathematics of hardware reliability theory to the prediction of software reliability. Most hardware related reliability models are predicated on failure due to wear rather than failure due to design defects. In hardware, failures due to physical wear are more likely than a design related failure. Unfortunately, the opposite is true for the system. In fact, all system failures can be traced to design or implementation problem; wear does not enter into the picture.

There is still debate over the relationship between key concepts in hardware reliability and their applicability to system. Although an irrefutable link has yet to be established, it is worth while to consider a few simple concepts that apply to both system elements.

If we consider a computer-based system, a simple measure of reliability is mean time between failures (MTBF), where,

$$\text{MTBF} = \text{MTTF} + \text{MTTR}$$

(The acronyms MTTF and MTTR are mean time to failure and mean time to repair, respectively.)

Many researchers argue that MTBF is a far more useful measure than defects /KLOC. Stated simply, an end user is concerned with failures, not with the total defect court. Because each defect count provides little indication f the reliability of a system. For example, consider a program that has been in operation for 14 months. Many defects in this program may remain undetected for decades before they are discovered. The MTBF of such obscure defects might be 50 or even 100 years. Other defects, as yet undiscovered, might have a failure rate of 18 or 24 months. Even if every one of the first category of defects is removed, the impact on system reliability is negligible.

In addition to a reliability measure, we must develop a measure of availability. Software availability is the probability that a program is operating according to requirement at a given point in time and is defined as:

$$\text{Availability} = (\text{MTTF}/\text{CMTTF} + \text{MTTR}) \times 100\%$$

The MTBF reliability measure is equally sensitive to MTTF and MTTR. The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.

9.6 QUALITY ASSURANCE

The quality of an information system depends upon its design, development, testing and implementation. One aspect of systems quality is its reliability. A system is reliable if, when used in a reasonable manner, it does not produce failures that can be dangerous or costly. This definition distinguishes between software errors, instances of the system's not producing the expected result and failures as well as the occurrences of software errors. Although it is virtually impossible to develop software that can be proven to the error-free, yet software developers strive to prevent the occurrence of errors, using methods and techniques that include error detection and correction and error tolerance. Both these strategies are useful for keeping the system operating and preventing failure, of software. Failures are result of design errors that were introduced when specifications were formulation and software written.

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability. And, of course, it includes assurance that the system meets the specifications and the requirements for its intended usage and performance.

An additional aspect of quality assurance is avoiding the need for enhancement on the one hand and developing software that is maintainable on the other. The need for maintenance is very high and impedes new developments.

Quality assurance has been defined as a planned and systematic pattern of all the actions necessary to provide adequate confidence that software conforms to established technical requirements. The key ideas or goals of quality assurance are as follows:

1. ***Comprehensiveness:*** Quality assurance is not restricted to the function of a software quality groups or phase. It includes all the necessary activities that contribute towards the quality of software through out the entire life-cycle of a project.

2. ***Planning:*** The emphasis is on a systematic plan to achieve the objectives of software quality. The quality of a piece of software is not left to the efforts of individuals.

3. ***Relativity:*** The notion of quality is relative to some requirements. The purpose of quality assurance is not to guarantee 100% reliability or zero detect software. It is rather to increase confidence that every reasonable effort has been made to ensure the quality of the end product. Quality, therefore, equals conformance to requirements, not excellence.

4. ***Cost:*** While purchasing any product, the level of quality of a product is usually reflected in its price. Hence, software quality involves increased cost. Product quality is, therefore, wholly a matter of customer choice. It is essential at the requirements analysis stage for the analyst to identify appropriate customer quality needs.

Preparation of a software quality assurance plan for each software project is a primary responsibility of the software quality group.

Topics in a software quality assurance plan include the following:

1. Purpose and scope of the plan.
2. Documents referred in the plan.
3. Organisational structure, tasks to be performed and specific responsibilities as they relate to product quality.
4. Documents to be prepared and checks to be made for the adequacy of the documentations.
5. Standards, practices and conventions to be used.
6. Reviews and audits to be conducted.
7. A Configuration management as well as plan that identifies software product items, controls and implements changes and records reports change status.
8. Practices and procedures to be followed in reporting, tracking and resolving software problems.
9. Specific tools and techniques to be used to support quality assurance activities.
10. Methods and facilities to be used to maintains and store controlled versions of identified software.
11. Methods and facilities to be used to protect computer program physical media.
12. Provisions for ensuring the quality of vendor - provided and subcontractor-developed software.
13. Methods and facilities to be used in collecting, maintaining and relating quality assurance records.

Other duties performed by quality assurance personnel include the following:

1. Development of standard policies, practices and procedures.
2. Development of testing tools and other quality assurance aids.
3. Performance of the quality assurance functions described in the software quality assurance plan for each project.
4. Performance and documentation of final product acceptance tests for each software product.

More specifically, a software quality assurance group may perform the following functions:

1. During analysis and design, a software verification plan and an acceptance test plan all prepared. The verification plan describes the methods to be used in verifying that the requirements are satisfied by the design documents and that the source code is consistent with the requirements, specification and design documentation. The source-code test plan is an important component of the software verification plan. The acceptance test plan includes test cases, expected outcomes and capabilities demonstrated by each test case. Often, quality assurance personnel would work with the customer to develop a single acceptance test plan.
2. During product evolution, in-process audits are conducted to verify consistency and completeness of the work products. Items to be audited for consistency include interface specifications for hardware, software and people internal design versus functional specifications source code versus design documentation and functional requirements versus test descriptions.

3. Prior to product delivery, a functional audit and a physical audit are performed. The functional audit reconfirms that all requirements have been met. The physical audit verifies that the source code and all associated documents are complete, internally consistent and consistent with one another and ready for delivery. A software verification summary is prepared to describe the results of all reviews, audits and tests conducted by quality assurance personnel throughout the development cycle.

Quality assurance personnel are sometimes in charge of arrangements for walk throughs, inspections and major milestone reviews. In addition, quality assurance personnel often conduct the project's postmortem, write the project legacy document and provide long-term retention of project records. The quality assurance organisation can also serve as a focal point for collection, analysis and dissemination of quantitative data concerning cost distribution, schedule slippage, error rates and other factors that influence quality and productivity.

9.7 QUALITY ASSURANCE AND THE ORGANISATION

The quality ethic has become influential in business circles ever since the Japanese demonstrated the marketability of the concept. In organisations, where primary business is the production of information systems, the question of quality is influenced by a number of issues as follows:

1. ***Project Management:*** It is argued that an effective project manager would build, into his monitoring process, clear checkpoints for assessing the quality of a developing information system.
2. ***Case:*** Many argue that case tools offer the developer a route to better quality systems because fewer errors are likely to occur between states such as design and implementation.
3. ***Formal Methods:*** The advocates of formal methods would claim that approach offer a key to quality be and implementation can proved to meet its specification.
4. ***Object-orientation:*** Object-oriented languages foster the reuse of coding of an information system from well proven parts and are likely to lead to increase quality.

9.8 SYSTEM AUDIT

In any business, it is obligatory to conduct the audit of the business records and documents. The auditors study documents like ledgers, balance sheets, etc., to see if the documents have been prepared correctly and the entire lot of transactions have been properly accounted for. In accounting, a financial transaction is the basic entity. In a computer system, a transaction is the smallest entity. A transaction could be reading a record, writing a record, putting a value in a field, adding two numbers, printing a report, input of value, etc.

In system audit, the auditors study these transactions to see that they have been properly performed or not. Auditing is a part of testing. It usually involves keeping log of transactions. But auditing may also be possible for online users to sign onto a system, alter data stored in the files and sign off again, without leaving a visible trace as to what happened. Unless the analyst develops an audit trail, no such protection exists in online and distributed systems.

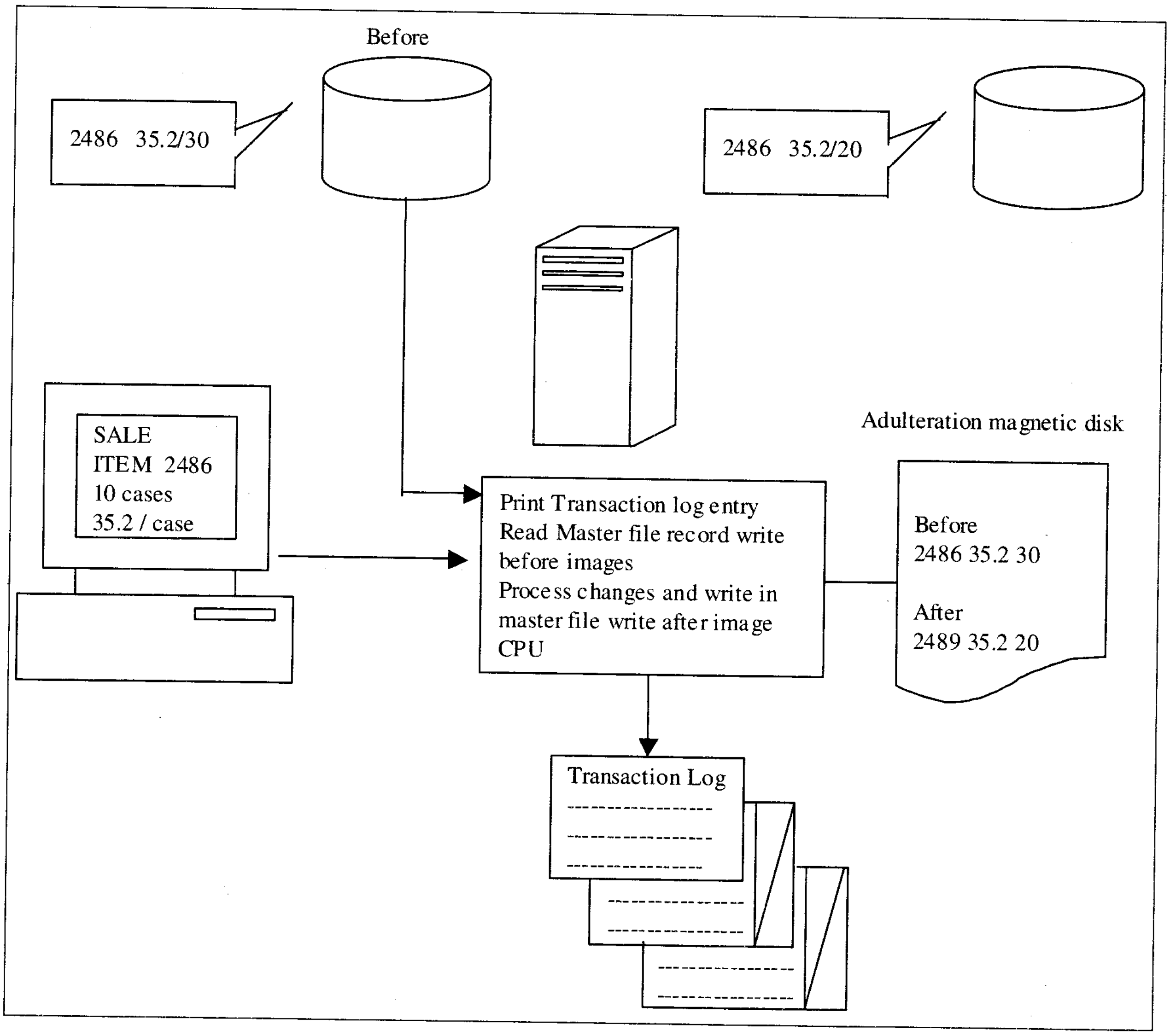

Before
2486 35.2/30
2486 35.2/20
SALE
ITEM 2486
10 cases
35.2 / case
Adulteration magnetic disk
Print Transaction log entry
Read Master file record write before images
Process changes and write in master file write after image
CPU
Before
2486 35.2 30
After
2489 35.2 20
Transaction Log

Figure 9.1: Audit Trail

An audit trail is designed to permit tracing of any input record process performed on a system back to its original source. One way of accomplishing this is by automatically maintaining a transaction log. The details of each transaction are recorded in a separate transaction file on the system (Figure 9.1). Before and after images can provide information on how the record was changed. If this record has been changed illegally, the name of the user could be found out by the system administrator since the date, time and location (terminal) of transaction may also be recorded.

The storage of these details is automatic and invisible to the user about whom information should also be stored so that it is clear who conducted the transaction. If the system has an internal clock, each transaction is also time stamped to tell when it occurred. If the need arises to audit a particular record in a file, it is relatively easy to determine who submitted the transaction, when it occurred, what data the transaction contained and what modifications were made in the master file. In other words, it is a complete trail of the entire transaction and its effect on the system.

Another form of audit trail presumes that keeping transaction data on magnetic disk is not fully reliable. For example, in some small business systems, if the system is turned off, perhaps through a power failure, before the data captured on disk during an editing session has been backed up, it will be lost completely (Some computer systems use disk systems in which the read-write heads drop down to the disk surface when the power is turned off. In these systems, users must remove the disk before powering down or they will loose data).

Printing a copy of the transaction before processing it is one of the best ways to protect data against losses. Then if anything happens during the online session, a backup copy of the master file can be mounted and transactions re-entered by using the printed transaction list.

Users many miles away have no way of learning when a malfunction occurs. As long as they can continue to enter transactions, they have every reason to believe that the system is operating correctly. Therefore, the analyst must anticipate these problems for safeguarding the integrity of a system by providing ways to audit it's use.

9.9 WHAT IS SYSTEMS SECURITY?

The protection of computer-based resources which include hardware, software, data, procedures and people against unauthorized use or natural disaster is known as security or system security.

Contrary to system security, which concerns with the whole system, data security refers to the protection of data from modifications, deletions, destruction or disclosure by unauthorised persons.

9.10 NEEDS OF SECURITY

The computer and information systems represent valuable assets of the organisation. They are required to be protected against unauthorised access, fraud, embezzlement, fires and natural disasters. Without providing proper safeguards to computer-based resources, an organisation cannot survive. Security of PCs and information systems is mandatory for every organisation. Security is needed mainly due to the following reasons:

1. ***Privacy:*** Privacy defines the rights of the individual users or organisations to keep their personal data and information secret. Most people feel that their personal information should not be monitored by others without their consent. However, the present information technology enables people to monitor and collect personal data without the individual's consent. Without security, the important confidential data of organisations is often sold or sent to other companies and is misused. Mailing lists of customers is also sold or sent to the competitors of the organisation. Therefore, all individual users and organisations need security for maintaining the privacy of their confidential data and softwares.

2. ***Accuracy:*** Most of the damage to data and systems of an organisation is caused by errors and omissions made by people. An organisation always needs accurate data for processing transactions, providing better service to their customers and making effective decisions. As computerised organisations mainly rely on databases, it is most likely that employees change the data intentionally or accidentally. The incorrect data leads to erroneous results and conclusions. The customers always suffer due to erroneous statements. Therefore, all organisations need security to maintain the accuracy of data.

3. ***Threats by Dishonest Employees:*** A dishonest employee is a major threat to an organisation. A dishonest programmer can easily access any software or data, if proper control measures are not provided to the system. The dishonest employees can manipulate the confidential letters, financial data/statements or other official documents and can send to the competitors of the organisation. They can easily steal secrets from computers, which can be even more profitable than robbing banks. Therefore, security becomes mandatory for organisations to disable the dishonest employees for stealing official data from computers.

4. ***Threats of Fire and Natural Disasters:*** Fire and natural disasters like floods, storms, snowstorms, lightning, etc., are likely to cause excessive damage to the PCs, softwares and data. There should be adequate measures to protect computer-based resources against such disasters.

5. ***Computer Crimes:*** Computer resources can be misused for unauthorised or illegal functions, which are called as computer crimes. Computer crime includes embezzlling money by bank employees, unauthorised copying of costly softwares, unauthorised modifications of data to falsify important records, malfunctioning of computers/networks and performing other illegal functions for causing harm to computers. An organisation must provide adequate security measures for preventing computer crimes.

Forms of Computer Crimes

There are several forms of computer crimes which are described below:

1. ***Data Diddling:*** Data diddling refers to the unauthorised modification of data by dishonest employees of the organisation. Data diddling is mostly done by the staff. For instance, an employee may falsify his record by making modifications in the data. An accountant may change the amount figures of the debtors, which could cause excessive loss to the organisation. A bank employee performing reconciliation can embezzle millions of rupees by making modifications of data. He can also illegally transfer money to his account.

2. ***Trapdoor Programs:*** Trapdoors refer to the special routines of a program, which allows the programmer to monitor storage area of memory (such as memory addresses) or other technical details of the system in order to check whether the program is performing correctly or not. These routines (if available in the system) allow the users to illegally access passwords (secret codes) and other important information. Although, trapdoors are used to judge the performance of the system, they may be misused for making computer crimes.

3. ***Superzap Programs:*** Superzaps refer to another special routine of the software, which do not have regular control mechanisms and passwords. These are used only when the normal functioning of the software is disturbed or stopped. Although, only authorised persons can use superzap programs, there is a possibility that unauthorised persons may use them. In those circumstances, superzap programs may lead to computer crimes.

4. ***Logic Bombs and Viruses:*** Logic bombs are the most dangerous computer crimes. Logic bombs refer to special routines that cause excessive damage to the data or vital programs (such as operating system files) and thus either disturb or completely stop the working of the system. For example, a logic bomb may be a program:

   ❖ To format the hard disk;

   ❖ To erase system files;

- To delete important files on a particular date;
- To change important data; or
- To copy itself to other programs alongwith creating many problems to the system, which is called a virus.

Viruses are the most common form of logic bombs. Viruses refer to the programs that invade and disrupt the normal working of PCs by spreading from one PC to another. A virus spreads like a biological virus as discussed below:

(i) A virus attacks when someone writes or copy a program that embeds itself in a host program.

(ii) The virus attaches itself to the host program and travels from one PC to another either through floppy disk or data communications networks.

(iii) A virus may cause minor or severe damage to the data or create unexpected problems in the infected PCs.

5. ***Hacking:*** Hacking refers to get into someone else's computer system without permission in order to find out information or do something illegal. Hacking has become very widespread over the last decade. A person who gains access to a computer system or network without authorisation is called as hacker. Hackers generally write programs in assembly or C language and break into a computer system just for accepting the challenge of doing so. For instance, in April 1992, some hackers invaded the system of a US credit bureaus company and stole credit history records of numerous individuals.

6. ***Leakage:*** Leakage refers to unauthorised copying of confidential data from hard disk to floppy disks. Leakage results when an employee leaves the office by carrying sensitive data on small floppies, which cannot be detected by security persons. The leakage of sensitive data enhances computer crimes.

7. ***Eavesdropping and Wiretapping:*** Eavesdropping refers to monitoring of data transmissions which are meant for other persons. Wiretapping is one of the methods of eavesdropping. Wiretapping refers to setting up a special transmission path for diverting the flow of data. Wiretappers generally attach a cable on the local area network for diverting the flow of transmitted data.

   Eavesdropping on Internet: As an access to the internet cannot be easily restricted, any information that is transmitted on the Internet is subject to eavesdropping. This is because of the following reasons:

   (a) On Internet, the information is passed through multiple sites and telecommunication links before reaching the target destination. So, anybody on the way can easily eavesdrop the information.

   (b) Most of the hosts of the Internet are Unix-based, which is notoriously weak on security. Thus, eavesdropping on Internet is easier.

8. ***Downloading:*** Downloading refers to receiving of data and software, which are transmitted from other computers. Download has become a common method of copying files via Internet. So, Internet helps computer users in unauthorised copying of data and programs from all over the world and thus provides a convenient way for making computer crimes.

9. ***Software Piracy:*** Software piracy refers to illegal copying of software for the purpose of distribution or sale. It is the most common and widely spread form of computer crime. This is because that most users prefer to use pirated copies rather than spending money on buying softwares. Software piracy is the most serious problem of today's software industry.

   Software Piracy in India: For many years, the Indian software market has been dominated by pirated software. It is a very serious computer crime which is done by 99% Indian users. The copies of all latest software such as Windows, MS Office, PageMaker, CorelDRAW, Oracle, etc., costing lakhs of rupees are available on pirated CDs or Floppy Disks in just few hundreds of rupees or even sometimes free of cost. The users want to invest only in hardware and expect free copies of software with every PC. Hardware vendors also promote software piracy because they understand that they will be unable to sell a PC in India, unless they offer few software free of cost with every PC. Actually in India, the user mentally sets the value of software simply as the cost of a floppy disk. This is because that the legal software are very costly in India as compared to hardware. Although, software piracy has become the greatest threat to software industry, it does harm the security of an organisation because it is the data which is important for an organisation and not the software.

9.11 NEEDS OF SYSTEM CONTROL

Besides security aspects, adequate control measures are needed in the system due to the following reasons:

(a) ***Reliability:*** During development of the system, the main objective of the analyst is to ensure the reliability of the system. Reliability is the main criteria for quality assurance of the software.

(b) ***Efficiency:*** Efficiency refers to the numbers volume and capacity of computer resources (hardware storage devices, memory, etc.) used by the system during execution. Adequate control measures must be built in the system to make it efficient.

(c) ***Correctness:*** The system must meet the user objectives and software requirements specification. Security control is required to make the system correct as per the specifications given by the user.

(d) ***Accuracy:*** Inaccurate data leads to inaccurate results and outputs. The system must have adequate controls to ensure the accuracy while feeding the data.

(e) ***Usability:*** Graphical user interface is the basic need of present day systems. The system must be easy to operate and learn. Usability is the another need of system control.

(f) ***Maintainability:*** The system must be easy to modify and maintain. System control ensures the maintainability of the system.

9.12 SYSTEM CONTROL AND SECURITY MEASURES

We have discussed the importance of security for an organisation and an individual's PC. Now, we will emphasise on various security measures. Before discussing these measures, let us first see the difference between the following terms:

- ***System Integrity:*** System integrity refers to the proper functioning of computer components (hardware/software) by providing physical security and safety measures against external threats.

- ***Data Integrity:*** Contrary to system integrity, data integrity refers to surety of the completeness and consistency of data by providing appropriate security measures such as data validation. Data validation refers to the measures taken to ensure the validity, completeness and consistency of data.

The organisation must follows the following control measures for providing adequate security to PCs and information systems:

(a) Physical Security

(b) Identification of Users

(c) Access Control

(d) Encryption and Decryption

(e) Audit Controls

Physical Security

Physical security includes safeguards against damage of hardware, software and data due to the physical environment of the PCs. The computer room and data library are the most critical areas of physical security. The major threats to the physical security are:

- Destruction of hardware
- Loss of documentation
- Damage of databases
- Fire
- Water
- Theft of computer-based resources
- Sabotage (to prevent the success of computerisation by destroying computer equipments)
- Loss of power

The organisation must take following preventive measures against the above threats:

- The entry and exit to computer room and data library should be restricted and properly monitored by closed circuit televisions and security guards.
- The hardware, software and data must be protected against theft by installing locked doors, theft-proof safes and vaults, burglar alarms and closed-circuit televisions in the computer room and data library.
- Sensors for early detection of fire, fireproof vaults and wall-mounted fire-extinguishing systems must be installed in the entire office building.
- Water pipes should be located away from the computer room, data library and communication lines.
- Ceilings and walls should be properly sealed against water seepage.
- External storage devices having backups of data and software should be stored either away from the computer rooms and data library in the same building or away from the office building.

- Sprinklers (equipments for scattering water to put out fires) should be fitted in the office premises.
- Fibre optic cables, which are relatively safe from wiretaps, should be used in cabling of local area networks.

Identification of Users

The authorised users must be identified before using computer resources. The identity of users may be established by one or more of the following methods:

1. ***Using Password:*** Password is a group of characters, used to access certain hardware, software or data. A password is the most commonly used method for verifying the identity of user. A password should have the following characteristics:

   (i) It should be a meaningful word or phrase that can be remembered easily. For instance, 'DELHI', 'YEAR2000', 'I LOVEYOU', 'GOODDAY', etc., are some of the meaningful passwords.

   (ii) It should not be related with the system to be accessed. For instance, for using the payroll system, the passwords 'PAY', 'SALARY', 'PAYMAKER', 'BASICPAY', etc., can be guessed by other persons. Therefore, a password must be easy to remember but hard to guess.

   (iii) It should not be the name of user who is using the system or the name/place of company where it is installed. For instance, if the user named 'Dr Suchitra' is authorised to use a system of company named 'National Computer Lab (Regd.)' located at 'Vikas Puri, Delhi', the passwords 'DOCTOR', 'SUCHITRA', 'NCL', 'NATIONAL','VIKASPURI', 'DELHI', etc., can be easily guessed.

   (iv) Never use the passwords consisting of only numbers. For instance, '12345', '1998', '2000', '39230', etc., may be difficult to remember but easier to guess.

   (v) Never write the passwords on any paper or in the computer file because it may be leaked out.

   (vi) Use only unrelated word or phrases as passwords which nobody could think about. For instance, using a 'Financial Accounting System', the passwords 'CRICKET', 'BUTTERFLY', 'CAT', 'I LOVEYOU', etc., are extremely difficult to guess.

   Change your passwords regularly.

   Are the Passwords really secure?

   Passwords are not always 100 per cent secure because they may be:

   - leaked out;
   - forgotten by the user;
   - guessed by others even if the chances are less than 1 per cent;
   - find out by experts having knowledge of systems programming;
   - searched by computer utilities.

2. ***Using Fingerprints:*** Fingerprints can also be used for identity of users in an organisation by using the latest technology. Although, it is the best method for identifying the users, it is not used due to non-availability of the required hardware/software in many organisations. Currently, fingerprints technology is only used in law enforcements.

3. ***Using Access Cards:*** In some organisations, access cards are provided to authorised users. Besides knowing password, the user must possess access card in order to use the system. For instance, access cards are used in automated teller machines for withdrawing money.

4. ***Using Keys:*** Users are provided keys to unlock their terminals. The user is identified on the basis of the key.

5. ***Restricted System Use:*** Users are allowed to use only those terminals and resources to which they are authorised. The restricted use of system makes the identification of users easier.

Access Control

The access to various computer resources by authorised users must be properly controlled. The main guidelines to control access are:

- Provide a single entrance to the computer room monitored by security guards.
- Note down the name of the user, his ID No., terminal used, time and duration of use, purpose of use, etc., in the entry register.
- Use an encoded card system which serves as a key to unlock doors. The encoded card system consists of a magnetic key and a lockport. When card is inserted into the lockport, the door is unlocked.
- Employ a librarian to monitor the use of floppies and various hardware/ software resources by permitting authorised persons only.

Encryption and Decryption

Encryption is an effective and practical method to secure data. Encryption refers to encode data by converting the standard data code into a proprietary code. Just opposite to encryption, decryption is the process to convert encrypted data into its original form. Encryption and decryption techniques are commonly used during transmission of data from one computer to another. The basic concept of encryption/decryption process is illustrated in Figure9.2 and is explained:

(i) The plaintext message, which has to be transmitted, is encrypted to produce a ciphertext.

(ii) The ciphertext is transmitted to other terminals over communication lines.

(iii) The ciphertext is received at authorised receiver and is decrypted back into plaintext.

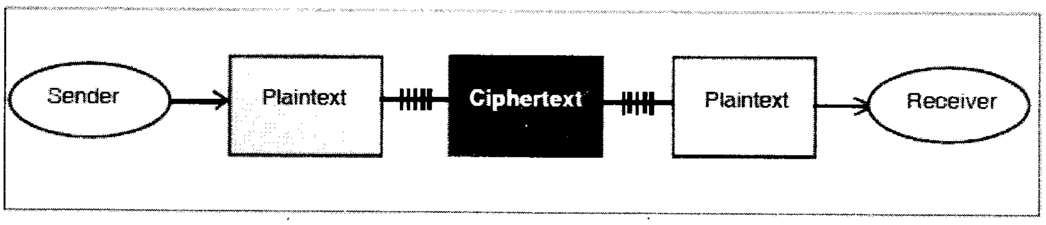


Figure 9.2: Concept of Encryption/Decryption

The general technique of encryption/decryption is based on algorithms given by National Bureau of Standards. This technique is known as Data Encryption Standard (DES). It is widely used in many network security systems. DES uses a binary number as the key for encryption. This key offers more than 72 quadrillion combinations. The binary number is used as a pattern to convert the bits at both the ends of transmission. For each transmission, the key can be changed randomly.

Encryption/Decryption Software

Data files can also be encrypted by using utility software. These software permit the user to encrypt and/or decrypt files in an interactive menu driven mode. These software encrypt a file by randomly selecting one of the encryption algorithms available. The encrypted files can be decrypted by authorised persons only due to use of password during encryption/decryption process.

Audit Control

Although, all the above discussed security measures are sufficient to protect the system, there may be some breaches in the security. Audit control protects the system from possible security breaches and frauds. Audit, with respect to systems, refers to examination of working systems (hardware/software) in order to determine their efficiency. Software implementation and maintenance processes must be audited properly to prevent any embezzlement. Various audit control software can also be used by auditors to examine databases for their correctness, completeness and consistency. The best way for audit control is that the application software should have inbuild audit control procedures so that the system can examine itself.

9.13 SECURITY ON NETWORKS

Security is critical for maintaining the integrity of data stored on the network servers. Security, with reference to a network, refers to operating system controls used by the network administrator to limit users' access to approved areas. Security is implemented in the network operating system at following levels:

(a) Login and Password Security

(b) Account Security

(c) Directory and File Rights Security

9.13.1 Login and Password Security

Login, with reference to networks, refers to establishing a connection to the server by an authorised user. It requires a password to access the system. The first level of network security is the password protection, which enables the user to use one or more passwords to prevent unauthorised access. To prevent an individual user from login to the network from several workstations, the number of concurrent connections can be limited. The timings of day, when a user can access network, can also be restricted.

9.13.2 Account Security

On local area networks, an account is set up for each user. A user account is used to control access to a network established and maintained by the network administration. The elements of a user account include the following details:

(i) Password information

(ii) Rights of the user

(iii) Information about groups to which the user belongs.

9.13.3 Directory and File Rights Security

Each user on network is given limited rights to access specific directory and files. The various types of file and directory rights available in LAN are summarised in Table 9.1.

Table 9.1

Level	Description
Class D	An operating system that is not secure (e.g. MS DOS).
Class C1	Requires an individual login, but allows group identification.
Class C2	Requires an individual level of login by password with an audit mechanism (e.g., Novel Netware 4.x, Windows NT Server).
Class B1	Requires Department of Defence Security clearance levels.
Class B2	Guarantees a path between the user and the security system and ensures that clearances cannot be changed.
Class B3	Security based on a mathematical model that must be viable and repeatable. The system must be managed by a network administrator in charge of security and must remain secure when shut down.
Class A1	The highest level of security based on a mathematical model that can be proven.

Security Levels

The United States Department of Defense Standard specifies the security levels for various operating systems and networks which are summarised in Table 9.2.

Table 9.2

Right	Function
	Permits the user:
Open	To open an existing file in a sub-directory
Create	To create a new sub-directory or file
Read	To read the contents of a file
Write	To modify the contents of a file
Delete	To delete a file
Search	To list the files in a sub-directory
Modify	To change the attributes of a file
Parental	To read, rename and delete a sub-directory

9.14 DISASTER RECOVERY

Inspite of all security measures, organisations are always vulnerable to disaster. Disaster recovery refers to the contingency measures that an organisation must take to recover the data in case it is damaged or lost due to security breaches. Backup is the most efficient method of disaster recovery.

Backup

Backup, with reference to software and data, refers to creation of duplicate copies of important data and software on various storage devices, such as magnetic tapes and floppies in order to restore data in event of hardware or software failure. Backup is extremely important in a disaster recovery procedure. Backup, sometimes, is also defined with reference to hardware. Hardware backup involves keeping duplicate hardware components, especially computer peripherals (I/O and Storage devices) in case of their failure or malfunctioning. However, 'backup' term is generally understood as 'backup of data and software'.

Guidelines for Taking Backups

The basic guidelines for taking backup of data are as follows:

- Make a proper plan and follow a suitable strategy for backup and recovery procedure.
- Keep backup of data files regularly, preferably on a daily basis.
- Keep multiple copies of backup (minimum of two copies).
- Store backup floppies or tapes away from the data processing centre.
- Use only genuine floppies for taking backups.

9.15 COMPUTER ETHICS

Standards of moral conduct to be followed by employees for using various computer resources are called as Computer Ethics.

Besides all the discussed security measures, ethics are also required to maintain the system security. Although, unethical acts are not always illegal, they may cause harm to the security of the organisation. Unethical acts may be performed by both employees and people outside organisations.

Code of Computer Ethics

Organisations develop a code of ethics in order to create awareness among employees to understand their duties. Employees must know the difference between right and wrong in order to be ethical. Some of the guidelines which are generally given in a code of computer ethics are described below:

- All employees will maintain the privacy and confidentiality of the organisation's data and official documents.
- No employee without proper authorisation, shall make changes in the data in any way form.
- Employees will not carry with them floppies having official data while leaving office.
- No employee, for whatever reason, shall give any floppy having official data and legal software to any outsider.
- No employee shall be involved in any conduct related to software piracy.
- All employees will use the software in accordance with the license agreement between purchaser and seller.
- No employee shall access the data meant for others.
- Employees will not use knowledge of a confidential nature for their personal use.

Check Your Progress

1. Fill in the blanks:
   (a) Fire and water are the major threats to .... security.
   (b) .... is an effective and practical method to secure data.
   (c) The first level of network security is the .... protection.
   (d) .... is the most efficient method of disaster recovery.
   (e) Organisations maintain a .... in order to create awareness among employees to understand their duties.
2. State true or false:
   (a) Floppies having backups of data should be stored in the computer room in order to provide physical security to the organisation.
   (b) Access cards are used to identify the user.
   (c) A password should be related with the system to access.
   (d) Audit controls protects the system from possible security breaches and frauds.
   (e) If all security measures are followed, organisations are not vulnerable to disasters.
3. Match the following:

(a)	System Integrity	(i)	Destroying Computer Equipments
(b)	Encryption	(ii)	Security Level
(c)	Data Validation	(iii)	Providing Physical Security
(d)	Sabotage	(iv)	Ciphertext
(e)	Account Security	(v)	Identification of Users
(f)	Class B1	(vi)	Duplicate Copies
(g)	Fingerprints	(vii)	Security on Networks
(h)	Backup	(viii)	Data Integrity
(i)	Password	(ix)	Automated Teller Machines
(j)	Access cards	(x)	Law Enforcements

9.16 LET US SUM UP

Quality control and assurance is an integral part of all production and development processes. System control governs all the activities of the software development process. System security refers to the protection of computer-based resources system reliability is the degree to which the system performs its intended functions over a span of time.

Security of PCs and information system is needed due to privacy, accuracy, threat from dishonest employees, threats from fire and natural disasters and computer crimes. Various forms of computer crimes include data diddling, trapdoor progra·ns, superzap programs, logic bombs and viruses, hacking, leakage, eavesdropping and wiretapping, downloading and software piracy.

Besides security aspects, adequate control measures are needed in the system due to reliability, efficiency, correctness, accuracy, usability and maintainability of the system. The organization must follow control measures for providing adequate security to PCs. The authorized users must be identified before using computer resources. The access to various computer resources by authorized users must also be properly controlled.

Software implementation and maintenance processes must be audited properly to prevent any embezzlement. Audit trail is designed to permit tracing of any input record process performed on a system back to its original source. Another form of audit trail presumes that keeping transaction data o magnetic disk is not fully reliable.

Security on networks refers to operating system controls used by the network administrator to limit users' access to approved areas. The United States Department of Defense standard specifies the various security levels for various operating systems from class A1 to Class D.

Inspite of all security measures, organizations are always vulnerable to disaster. Disaster recovery refers to the contingency measures that an organization must take to recover the data incase it is damaged or lost due to security breaches. Backup of data and software is extremely important in a disaster recovery procedure.

Computer Ethic are the standards of moral conduct to be followed by employees for using various computer resources. Employees must know the difference between right and wrong in order to be ethical.

9.17 KEYWORDS

Reliability: The degree to which the system performs its intended functions over a span of time.

System Security: The protection of computer-based resources which include hardware, software, data procedures and people against unauthorized use or natural disaster.

Data Security: The protection of data from modifications, deletions, destruction or disclosure by unauthorized persons.

Privacy: The rights of the individual users or organizations to keep their personal data and information secret.

Comuputer Crimes: Misuse of computer resources for unauthorized or illegal functions.

Data Diddling: Unauthorized modification of data by dishonest employees of the organization.

Trapdoor Programs: Special routines of a program, which allows the programmer to monitor storage area of memory or other technical details of the system in order to check whether the program is performing correctly or not.

Superzap Programs: Specialroutine of the software, who do not have regular control mechanisms and password.

Logic Bombs: Special routines that cause excessive damage to the data or vital programs (Such as operating system files) and thus either disturb or completely stop the working of the system.

Viruses: Programs that invade and disrupt the normal working of PCs by spreading from one PC to another.

Hacking: To get into someone else's computer without permission in order to find out information or do something illegal.

Leakage: Unauthorized copying of confidential data from hard disk to floppy disks.

Eavesdropping: Monitoring of data transmissions which are meant for other persons.

Wire Tapping: Setting up a special transmission path for diverting the flow of data.

Downloading: Receiving of data and software, which are transmitted from other computers.

Software Piracy: Illegal copying of software for the purpose of distribution or sale.

Physical Security: Safeguards against damage of hardware, software and data due to the physical environment of the PCs.

Encryption: The process to encode data by converting the standard data code into a proprietary code.

Decryption: The process to convert encrypted data into its original form.

System Audit: Examination of working systems (hardware/software) to determine their efficiency. Auditing is a part of testing.

Disaster Recovery: The contingency measures that an organization must take to recover the data in case it is damaged or lost due to security breaches.

Backup: Creation of duplicate copies of important data and software on various storage devices to restore data in case of hardware or software failure.

Computer Ethic: Standards of moral conduct to be followed by employees for using various computer resources.

9.18 QUESTIONS FOR DISCUSSION

1. What is System Control?
2. Define System reliability. Give a simple measure of reliability.
3. Define quality assurance. Describe the key ideas of quality assurance.
4. What are the functions of a quality assurance group?
5. What are trapdoor programs? What do you mean by hacking?
6. Why do we need system control?
7. List various control measures for providing adequate security to PCs and information systems.
8. Define audit control.
9. Define security with reference to information systems. Why does an organization need security? Discuss in brief.
10. What is a computer crime? What are its different forms? Explain with suitable examples.

11. 'Any information that is transmitted on the Internet is subject to eavesdropping'. Do you agree with this statement? Discuss.
12. What is software piracy? Write a short note on software piracy in India.
13. Why does a PC is vulnerable to most computer crimes? Discuss.
14. Define the following terms:
    (a) Privacy
    (b) Data security
    (c) Hacking
    (d) Downloading
15. What are the major control measures for providing adequate security to PCs? Explain in brief.
16. What is physical security? What are the major threats to physical security?
17. What are the different methods to identify the users? Which one of them is the best and why?
18. Explain with examples the characteristics of a good password.
19. Describe the concept of encryption and decryption.
20. Discuss the various levels of security on local area networks.

Check Your Progress: Model Answers

1. (a) Physical
   (b) Encryption
   (c) Login and Password
   (d) Backup
   (e) Code of ethics
2. (a) False
   (b) True
   (c) False
   (d) True
   (e) True
3. (a) (iii)
   (b) (iv)
   (c) (viii)
   (d) (i)
   (e) (vii)
   (f) (ii)

Contd....

(g)	(x)
(h)	(vi)
(i)	(v)
(j)	(ix)

9.19 SUGGESTED READINGS

James C Wetherbe, St. Paul , *Systems analysis and design,* West Pub. Co., ©1988.

Jerry Zeyu Gao, H. S. Jacob Tsao, Ye. Wu, *Testing and Quality Assurance for Component-Based Software,* Artech House.

Pankoj Jalote , *An Integrated Approach to Software Engineering,* Springer.