NATIONAL OPEN UNIVERSITY OF NIGERIA

SCHOOL OF SCIENCE AND TECH.

COURSE CODE:-CIT 851

COURSE TITLE:- ADVANCED SYSTEMS ANALYSIS AND DESIGN

COURSE GUIDE

CIT 851
ADVANCED SYSTEMS ANALYSIS AND DESIGN

Course Developer/Writer — Vivian Nwaocha
National Open University of Nigeria

Programme Leader — Prof. Afolabi Adebanjo
National Open University of Nigeria

Course Coordinator — Vivian Nwaocha
National Open University of Nigeria

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
14/16 Ahmadu Bello Way
Victoria Island
Lagos

Abuja Office
No. 5 Dar es Salaam Street
Off Aminu Kano Crescent
Wuse II, Abuja
Nigeria

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Published by
National Open University of Nigeria

Printed 2009

ISBN: 978-058-678-4

CONTENTS **PAGE**

Introduction

The course, Systems Analysis and Design, is a core course for students studying towards acquiring the Master of Science in Information Technology. In this course we will study about the Systems Analyst as a

key role in system development. Various principles of design as well as different modeling techniques are discussed in this course. This course also considers various threats encountered by computer systems and various categories of information systems used in organisations.

The overall aims of this course are to introduce you to various ways of developing systems through the system development life cycle (SDLC). Implementation, maintenance and security issues are equally discussed.

In structuring this course, we commence with the basic process of system development and move to the actual implementation and maintenance of the system developed.

There are three modules in this course, each module consists of 10 units of topics that you are expected to complete in 3 hours. The three modules and their units are listed below.

What You Will Learn in this Course

The overall aims and objectives of this course provide guidance on what you should be achieving in the course of your studies. Each unit also has its own unit objectives which state specifically what you should be achieving in the corresponding unit. To evaluate your progress continuously, you are expected to refer to the overall course aims and objectives as well as the corresponding unit objectives upon the completion of each.

Course Aims

The overall aims and objectives of this course will help you to:

1. Develop your knowledge and understanding of the underlying principles of systems analysis
2. Build up your capacity to evaluate design techniques
3. Develop your competence in designing forms and reports
4. Build up your capacity to implement and maintain systems

Course Objectives

Upon completion of the course, you should be able to:

1. Describe the basic concepts of systems analysis

2. Explain the importance of a Systems Analyst
3. Identify the role of a Systems Analyst
4. Describe the process of designing a new system
5. Discuss the concepts related to documentation of systems
6. Explain the various design techniques
7. Describe the design of physical files and databases
8. Know how to design forms and reports
9. Describe systems implementation and maintenance
10. Discuss the audit and security of computer systems

Working through this Course

We designed this course in a systematic way, so you need to work through it from Module one, Unit 1 through to Module three, Unit 10. This will enable you appreciate the course better.

Course Materials

Basically, we made use of textbooks and online materials. You are expected to I, search for more literature and web references for further understanding. Each unit i has references and web references that were used to develop them.

Online Materials

Feel free to refer to the web sites provided for all the online reference materials required in this course.

The website is designed to integrate with the print-based course materials. The structure follows the structure of the units and all the reading and activity numbers are the same in both media.

Study Units

Module 1 Fundamentals of Systems Analysis and Design

Unit 1 Introduction to Systems Analysis and Design
Unit 2 Fundamentals of Systems
Unit 3 Systems
Unit 4 Approaches for Development of Information Systems
Unit 5 Systems Analyst -A Profession
Unit 6 Systems Analyst -Role, Duties and Qualification
Unit 7 Process of System Development
Unit 8 System Development Life Cycle
Unit 9 Documentation of Systems

From the preceding, the content of the course can be divided into three major blocks:

1. Fundamentals of Systems Analysis and Design
2. Planning and Designing Information Systems
3. System Design, Implementation and Maintenance

Module one describes the fundamentals of systems analysis and design.
Module Two explains the basic concepts of planning and designing information systems.
Module Three discusses the design, implementation and maintenance of systems.

Equipment

In order to get the most from this course, it is essential that you make use of a computer system which has internet access.

Recommended System Specifications:

Processor

2.0 GHZ Intel compatible processor
1GB RAM
80 GB hard drive with 5 GB free disk
CD-RW drive.
3.5" Floppy Disk Drive
TCP/IP (installed)

Operating System

Windows XP Professional (Service Pack
Microsoft office 2007
Norton Antivirus

Monitor*

19-inch
1024 X 768 resolution
16-bit high color
*Non Standard resolutions (for example, some laptops) are not supported.

Hardware

Open Serial Port (for scanner)
120W Speakers
Mouse + pad
Windows keyboard
Laser printer

Hardware is constantly changing and improving, causing older technology to become obsolete. An investment in newer, more efficient technology will more than pay for itself in improved performance results.

If your system does not meet the recommended specifications, you may experience considerably slower processing when working in the application.
Systems that exceed the recommended specifications will provide better handling of database files and faster processing time, thereby significantly increasing your productivity.

Assessment

The course, Systems Analysis and Design entails attending a three-hour final examination which contributes 50% to your final grading. The final examination covers materials from all parts of the course with a style similar to the Tutor- marked assignments.

The examination aims at testing your ability to apply the knowledge you have learned throughout the course, rather than your ability to memorise the materials. In preparing for the examination, it is essential that you receive the activities and Tutor-marked assignments you have completed in each unit. The other 50% will account for all the TMA's at the end of each unit.

Tutor-Marked Assignment

About 20 hours of tutorials will be provided in support of this course. You will be notified of the dates, time and location for these tutorials, together with the name and phone number of your tutor as soon as you are allotted a tutorial group.

Your tutor will mark and comment on your assignments, keep a close watch on your progress and on any difficulties you might encounter and provide assistance to you during the course. You must mail your TMAs to your tutor well before the due date (at least two working days are required). They will be marked by your tutor and returned to you as soon as possible.

Do not hesitate to contact your tutor by phone, e-mail if you need help. The following might be circumstances in which you would find help necessary. You can also contact your tutor if:

you do not understand any part of the study units or the assigned readings
you have difficulty with the TMAs
you have a question or problem with your tutor's comments on an assignment or with the grading of an assignment

You should try your best to attend tutorials, since it is the only opportunity to have an interaction with your tutor and to ask questions which are answered instantly. You can raise any problem encountered in the course of your study. To gain maximum benefit from the course tutorials, you are advised to prepare a list of questions before attending the tutorial. You will learn a lot from participating in discussions actively.

Course Overview

This section proposes the number of weeks that you are expected to spend on the three modules comprising of 30 units and the assignments that follow each of the unit.

We recommend that each unit with its associated TMA is completed in one week, bringing your study period to a maximum of 30 weeks.

How to Get the Most from this Course

In order for you to learn various concepts in this course, it is essential to practice. Independent activities and case activities which are based on a particular scenario are presented in the units. The activities include open questions to promote discussion on the relevant topics, questions with standard answers and program demonstrations on the concepts. You may try to delve into each unit adopting the following steps:

1. read the study unit
2. read the textbook, printed or online references
3. perform the activities
4. participate in group discussions
5. complete the tutor-marked assignments
6. participate in online discussions

This course makes intensive use of materials on the world-wide web. Specific web address will be given for your reference. There are also optional readings in the units. You may wish to read these to extend your knowledge beyond the required materials. They will not be assessed.

Summary

The course, Systems Analysis and Design is intended to develop your understanding of the basic concepts of system analysis, thus enabling you acquire skills in designing new systems. This course also provides you with practical knowledge and hands-on experience in implementing and maintaining a system.

We hope that you will find the course enlightening and that you will find it both interesting and useful. In the longer term, we hope you will get acquainted with the National Open University of Nigeria and we wish you every success in your future.

Course Code	CIT 851
Course Title	Advanced Systems Analysis and Design
Course Developer/Writer	Vivian Nwaocha National Open University of Nigeria
Programme Leader	Prof. Afolabi Adebanjo National Open University of Nigeria
Course Coordinator	Vivian Nwaocha National Open University of Nigeria

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
14/16 Ahmadu Bello Way
Victoria Island
Lagos

Abuja Office
No. 5 Dar es Salaam Street
Off Aminu Kano Crescent
Wuse II, Abuja
Nigeria

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Published by
National Open University of Nigeria

Printed 2009

ISBN: 978-058-678-4

CONTENTS
PAGE

MODULE 1 FUNDAMENTALS OF SYSTEMS ANALYSIS AND DESIGN

UNIT 1 INTRODUCTION TO SYSTEMS ANALYSIS AND DESIGN

CONTENTS

1.0 INTRODUCTION

In this unit you will consider what systems analysis and design is and what it is not. You will also learn about the traditional systems development life cycle.

2.0 OBJECTIVES

After going through this unit, you should be able to:

explain what systems analysis and design is and what it is not
describe two major components of systems development
discuss the traditional systems development life cycle (SDLC).

3.0 MAIN CONTENT

3.1 What is Systems Analysis and Design?

In business, *systems analysis and design* refers to the process of examining a business situation with the intent of improving it through better procedures and methods.

3.2 Overview of Systems Analysis and Design

Systems development can generally be thought of as having two major components: systems analysis and systems design. *Systems design* is the process of planning a new business system or one to replace or complement an existing system. Before this planning can be done the old system must be thoroughly understood before we can determine how computers can best be used (if at all) to make its operation more effective. *Systems Analysis*, is then, the process of gathering and interpreting facts, diagnosing problems and using that information to recommend improvements to the system. This is the job of the systems analyst.

Systems Analysts do more than solve current problems. They are frequently called upon to help handle the planned expansion of a business. Analysts assess what the future needs of a business will be and what changes should be considered to meet those needs. Analysts will generally suggest a number of alternative solutions for improving the situation with recommendations as to which solution is the most applicable.

To summarise, analysis specifies what the system should do. Design states how to accomplish the objective.

3.3 What Systems Analysis is NOT?

Having looked at what systems analysis is -studying business systems to learn current methods and assess effectiveness. It may also be helpful to know what systems analysis is NOT:

It is NOT:

Studying a business to see which existing processes should be handled by computer and which should be done by non computerised methods.

The emphasis is on understanding the details of a situation and deciding whether improvement is desirable or feasible. The selection of computer and non computer methods is secondary.

It is NOT:

Determining what changes should be made.

The intent of the systems investigation is to study a business process and evaluate it. Sometimes, not only is change not needed, it is not possible. Change should be a result not an intent.

It is NOT:

Determining how best to solve an information systems problem.

Regardless of the organisation, the analyst works on business problems. It would be a mistake to distinguish between business and system problems, since there are no systems problems which are not first business problems. Any suggestions should be first considered in the light of whether it will improve or harm the business. Technically attractive ideas should not be pursued unless they will improve the business system.

3.4 The Traditional Systems Development Lifecycle (SDLC)

We begin by considering the period from the 1950s and covering the 1960s when there was no well-accepted formalised methodology to develop data processing systems Early uses of computing concentrated on scientific or mathematical applications. Later, when computers were installed in business environments, there were few practical guidelines which gave help on their use for commercial applications. These business applications were oriented towards the basic operational activities of the company and may include keeping files, producing reports and documents.

Typical examples would include:

customer records
sales records
invoice production
payroll

Applications which involve the basic data processing processes of copying, retrieving, filing, sorting, checking, analysing, calculating and communicating.

These early systems were implemented primarily by computer programmers who neither were not necessarily good communicators nor understood the user's requirements. Their main expertise lay in the

technological aspects of the systems. Standard practices and techniques were not in general use leading to an *ad hoc* approach to each project.

Problems associated with these developments were:

difficulties in the communication of user needs to system developers developments were frequently delivered late, over cost and did not meet the users needs

projects were viewed as short term solutions rather than long-term, well-planned implementation strategies for new applications.

changing the system was problematic and generally introduced new problems into the system. Therefore it appeared to take an inordinately long time to make relatively trivial changes.

documentation, if it existed, was usually out of date and programmers rarely had time to update it.

documentation standards were resisted by programmers as they were viewed as restricting creativity, increasing workloads and thus increasing overall development time. (true, but the time spent documenting a new system could pay dividends in reducing the maintenance time for systems).

companies were reliant on a few experienced programmers, new programmers found it difficult to take over because of the lack of documentation, uniform practices and techniques.

In answer to the problems outlined above the Software Development Lifecycle was adopted and adapted from a general development model common in other 'engineering' industries.

4.0 CONCLUSION

In this unit you have been introduced to systems analysis and design-what it is and what it is not. You have also been introduced to the concept of the systems development life cycle (SDLC).

5.0 SUMMARY

The lessons from this unit borders on the concept of systems analysis and design as well as the systems development life cycle (SDLC)

SELF ASSESSMENT EXERCISE

1. Describe two major components of systems development
2. List at least four problems associated with systems development

6.0 TUTOR-MARKED ASSIGNMENT

What do you understand by systems analysis and design?

7.0 REFERENCES/FURTHER READINGS

Valacich, Joseph *S.*, George, Joey F., and Hoffer, Jeffrey A. (2001) .*Essentials of Systems Analysis and Design*. Prentice Hall.

Kenneth E. Kendall, (2002). *Systems Analysis and Design*. Prentice Hall.

Edwards, Perry. (1993). *Systems Analysis and Design*. Mitchell McGraw-Hill.

Modell, Martin E. A Professional's Guide to Systems Analysis. (1996). *Introduction to the Tools and Techniques of Systems Analysis in the Business World*. McGraw-Hill.

Online Resources

http://www.raritanval.edu/departments/CIS/full-time/Schwarz/sad
http://www.abeuk.com/pdf/businessmanagement/SystemsAnalysisandDesign.pdf
http;//www.uml.org

UNIT 2 FUNDAMENTALS OF SYSTEM

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn about the fundamentals of system. You will equally consider the classification of systems and real life business subsystems.

2.0 OBJECTIVES

After going through this unit, you should be able to:

discuss the concepts related to systems
explain the classification of system
describe real life business subsystems.

3.0 MAIN CONTENT

3.1 Fundamentals of Systems

System is a word derived from the Greek word ‘Systema’ which means an organized relationship among components.

A System may be defined as orderly grouping of interdependent components linked together according to a plan to achieve a specific goal. Each component is a part of total system and it has to do its own share of work for the system to achieve the desired goal.

An **Information System** is an arrangement of people, data, processes, information presentation and information technology that interacts to support and improve day-to- day operations in a business as well as support the problem solving and decision making needs of management and users.

The characteristics of a System are as follows:

Organization implies structure and order. It is an arrangement of components that helps to achieve objectives.
Interaction refers to the procedure in which each component functions with other components of the system.
Interdependence means that one component of the system depends on another component.
Integration is concerned with how a system is tied together. It is more than sharing a physical part. It means that parts of system work together within the system even though each part performs a unique function.
Central Objective is quite common that an organization may set one objective and operate to achieve another. The important point is that the users must be aware about the central objective well in advance.

3.2 Important Terms Related to Systems

Purpose, Boundary, Environment, Inputs, and Outputs are some important terms related to Systems.

A System's **purpose** is the reason for its existence and the reference point for measuring its success.
A System's **boundary** defines what is inside the system and what is outside.
A System **Environment** is everything pertinent to the System that is outside of its boundaries.
A System's **Inputs** are the physical objects and information that cross the boundary to enter it from its environment.
A system's **Outputs** are the physical objects and information that go from the system into its environment.

3.3 Classification of Systems

Systems may be classified as follows:

a) Formal or Informal
b) Physical or Abstract
c) Open or Closed
d) Manual or Automated.

a) A **Formal System** is one that is planned in advance and is used according to schedule. In this system policies and procedures are documented well in advance. A real life example is to conduct a scheduled meeting at the end of every month in which agenda of the meeting has already been defined well in advance. An

Informal System is the system that is not described by procedures. It is not used.

According to a schedule. It works on as need basis. For example, Sales order processing system through telephone calls.

b) **Physical Systems** are tangible entities that may be static or dynamic.
Computer Systems, Vehicles, Buildings etc. are examples of physical systems. **Abstract systems** are conceptual entities.

Example: Company

c) **Open System** is a system within its environment. It receives input from environment and provides output to environment.
Example: Any real life system, Information System, Organization etc.

Closed System: It is isolated from environment influences. It operates on factors within the System itself. It is also defined as a System that includes a feedback loop, a control element and feedback performance standard.

Figure 1.1 shows a closed loop system. *Performance Standard* is defined as objective that the System has to meet. A *Feedback loop* is defined as a portion of the System that enables the System to regulate itself. Signals are obtained from the System describing the System Status and are transmitted to the Control Mechanism. A *Control Element* compares the output with the performance standard and adjusts the system input accordingly.

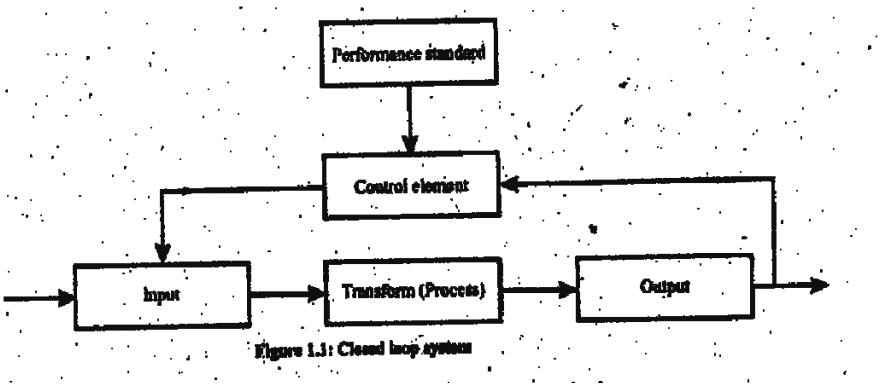


Figure 1.1: Closed loop system

d) **Manual and Automated Systems**: The system, which does not require human intervention is called Automated system. In this system, the whole process is automatic.

Example: Traffic control system for metropolitan cities.

The system, which requires human intervention, is called a **Manual System**. Example: Face to face information centre at places like Railway stations etc.

3.4 Real Life Business Subsystems

A Subsystem is a component of a System, even though it can also be considered as a system in its own right. Consider a manufacturing firm. It consists of five subsystems namely, Product design, Production, Sales, Delivery and Service.

The boundary is between the firth and its environment. In this system, all the subsystems work together to achieve a goal.

4.0 CONCLUSION

You would have learned about the concepts of related systems, classification of systems as well as real life business subsystems

5.0 SUMMARY

What you have learned in this unit borders on the fundamentals of systems.

SELF ASSESSMENT EXERCISE

1. Discuss the characteristics of systems?
2. Explain the real life business subsystems.

6.0 TUTOR-MARKED ASSIGNMENT

What are the characteristics of a system?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.rspa.com

UNIT 3 SYSTEMS

CONTENTS

1.0 INTRODUCTION

This unit introduces real times systems, describing the two types of real systems. Description of a distributed system is given and development of a successful system is explained.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

explain what a real time system is, listing the two types
describe a distributed system
discuss the development of a successful system.

3.0 MAIN CONTENT

3.1 Real Time Systems

A real time system describes an interactive processing system with severe time limitations. A real time system is used when there are rigid time requirements on the flow of data. A real time System is considered to function correctly only if it returns the correct result within imposed time constraints. There are two types of Real Time systems. They are:

Hard Real Time Systems which guarantee that critical tasks are completed on time.

Soft Real Time Systems which are less restrictive type of real time systems where a critical real time task gets priority over other tasks, and retains the priority until it completes them. Systems that control scientific experiments, medical imaging systems, industrial control systems and some display systems are real time systems.

3.2 Distributed Systems

A Distributed System in which the Data, Process, and Interface component of information System are distributed to multiple locations in a computer network.

Accordingly, the processing workload required to support these components is also distributed across multiple computers on the network. In this system, each processor has its own local memory. The processors communicate with one another through various communication lines, such as high buses or telephone lines. The processors in a distributed system may vary in size and function. They may include small microprocessors, workstations, minicomputers, and large general purpose computer systems. The implementation of a distributed system is complicated and difficult, but still is in demand. Some of the reasons are that modern businesses are already distributed. So, they need distributed solutions. In general, solutions developed using distributed systems paradigms are user-friendlier. They have the following advantages:

Resource sharing
Computation speedup
Reliability
Communication.

The five Layers of Distributed System architecture are:

Presentation Layer is the actual user interface. The inputs are received by this layer and the outputs are presented by this layer.
Presentation Logie layer includes processing required to establish user interface. Example: Editing input data, formatting output data.
Application Logic Layer includes all the logic and processing required to support the actual business application and rules Example: Calculations.
Data Manipulation Layer includes all the command and logic required to store and retrieve data to and from the database.
Data Layer is actual stored data in the database.

SELF ASSESSMENT EXERCISE 1

1. Define the following terms:

   Information system
   Strategic information
   Tactical information
   Operational information
   Repository

2. What is the difference between information requirements determination and specification?
3. What are the characteristics of a good information system?

3.2 Development of a Successful System

The success of any system depends on the approach of building it. If the development approach is right, the system will work successfully. Figure 1.2 depicts a System Development Life Cycle. System development life cycle (SDLC) is a standard methodology for the development of Information System. It mainly consists of four phases: System Analysis, System Design, System Construction & Implementation and System Support. Every phase consist of inputs, tasks and outputs. Traditional SDLC was strictly sequential. The developers first complete the previous phase then start the next phase. But now concept of Repository is introduced in SDLC and it is known as FAST methodology where work is done across shared repository. It means that all the inputs and outputs of phases must be stored in the repository. At any time developers can backtrack to previous phase and they can also work on two phases simultaneously.

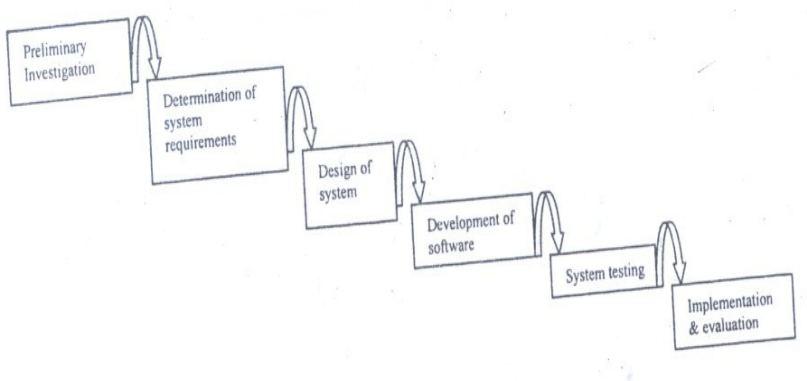


Figure 1.2: System Development Life Cycle

For making a successful system, the following principles should be followed:

(1) Both customers and developers should be involved for accuracy in the information.

(2) A problem solving approach should be adopted. The classic problem solving approach is as follows:

a) Study, understand the problem and its context
b) Define the requirements of a solution
c) Identify candidate solutions and select the best solution
d) Design and implement the solution
e) Observe and evaluate the solution's impact and refine the solution accordingly.

(3) Phases and activities should be established.
(4) For consistent development ofa system, some standards should be established.

These standards are:

Documentation Standards: It should be an ongoing activity during the system development life cycle.

Quality Standards: Checks should be established at every phase for ensuring that the output of every phase meets the business and technology expectations.

Automated Tool Standards: Hardware and software platforms should be finalized for the development of Information system. Automated tool standards prescribe technology that will be used to develop and maintain information systems and to ensure consistency, completeness, and quality.

(5) Development of information system should be considered as capital investment: The developer of an information system should think about several solutions of a particular problem and every solution should be evaluated for cost- effectiveness and risk management. *Cost-effectiveness* is defined as the result obtained by striking a balance between the cost of developing and operating an information system and the benefits derived from that system. Risk management is defined as the process of identifying, evaluating and controlling what might go wrong in a project before it becomes a threat to the successful completion of the project or implementation of the information system.

Multiple feasibility checkpoints should be built into system development methodology. At each feasibility checkpoint, all costs are considered sunk (i.e. not recoverable). Thus, the project should be re-evaluated at each checkpoint to determine if it remains feasible to continue investing time, effort, and resources. At each checkpoint, the developers should consider the following options:

Cancel the project if it is no longer feasible.
Re-evaluates and adjusts the cost and schedule if project scope is to be increased.
Reduce the scope if the project budget and schedule are frozen and not sufficient to cover all the project objectives.

Divide and Conquer approach is the way of making a complex problem easier. In this approach, the larger problem (System) is divided into smaller problems (Subsystem).

For development of a successful system, the system should be designed for growth and change. When the System is implemented, it enters the operations and support stage of Life Cycle (Please refer to figure 1.3).

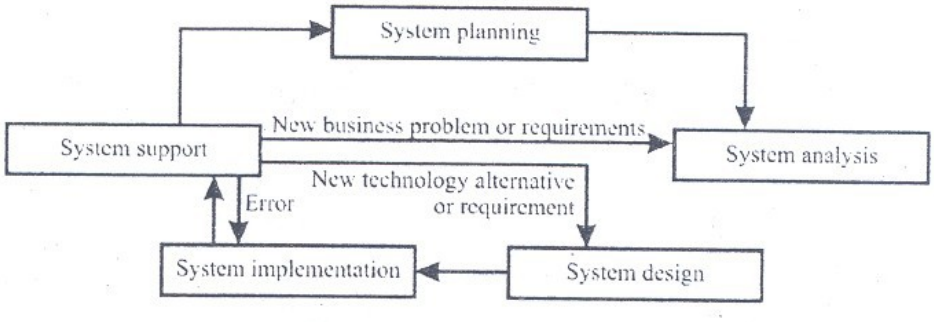


Figure 1.3: System Maintenance

During this stage, the developers encounter the need for changes that range from correcting simple mistakes to redesigning the system to accommodate changing technology to making modifications to support changing user requirements. These changes direct the developers to rework formerly completed phases of the life cycle.

4.0 CONCLUSION

Concepts about real and distributed systems were discussed in this unit. Standards for developing a system were equally discussed.

5.0 SUMMARY

What you learned in this unit borders on concepts of systems and standard methodology for the development of information systems.

SELF ASSESSMENT EXERCISE 2

1. Divide and conquer approach is a way of making a complex problem easier. True or false? Give reasons for your answer
2. Describe a real time system.

6.0 TUTOR-MARKED ASSIGNMENT

What principles should be adopted for making a successful system?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.rspa.com

UNIT 4 APPROACHES FOR DEVELOPMENT OF INFORMATION SYSTEM

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn about the various approaches for developing information systems. You will also learn about structured system analysis and design. Finally, you will learn about prototyping and joint application development.

2.0 OBJECTIVES

After going through this unit you should be able to:

describe the various approaches for developing information systems
understand the goal of structured systems analysis and design
explain the steps of prototyping process
describe the joint application development.

3.0 MAIN CONTENT

3.1 Various Approaches for Development of Information Systems

Various approaches are available for development of Information Systems. They are:

Model Driven: It emphasizes the drawing of pictorial system models to document and validate both existing and/or proposed systems. Ultimately, the system model becomes the blueprint for designing and constructing an improved system.

Accelerated approach: A prototyping approach emphasizes the construction of model of a system. Designing and building a scaled-down but functional version of the desired system is known as Prototyping. A prototype is a working system that is developed to test ideas and assumptions about the new system. It consists of working

software that accepts input, perform calculations, produces printed or display information or perform other meaningful activities.

Joint Application Development: It is defined as a structured approach in which users, managers and analysts work together for several days in a series of intensive meetings to specify or review system requirements. In this approach, requirements are identified and design details are finalized.

3.2 Structured Analysis and Design Approach

The goal of structured system analysis and design is tl) reduce maintenance time and effort. Modeling is the act of drawing one or more graphical representations of a System. Model driven development techniques emphasize the drawing of models to help visualize and analyze problems, define business requirements and design Information systems. The first model driven approach is Structured Analysis and Design approach.

Structured Analysis is a development method for the analysis of existing manual systems or automated systems, leading to development of specifications (expected functionality or behaviour) for proposed system. The objective of structured analysis approach is to organize the tasks associated with requirement determination to provide an accurate and complete understanding of a current situation. The major tasks of structured system analysis approach are:

Preliminary Investigation
Problem Analysis
Requirement Analysis
Decision Analysis.

It is a process-centred technique that is used to model business requirements for a system. Structured analysis introduced a process-modeling tool called the *Data flow diagram,* used to illustrate business process requirements. With the help of DFD, the systems analyst can show the system overview. Data modeling tools such as *Entity relationship diagrams* are used to illustrate business data requirements. With the help of ERD, the analyst, can show database overview.
Structured Design ‘utilizes graphic description (Output of system analysis) and focuses on development of software specifications. The goal of structured design is to lead to development of programs consisting of functionally independent modules that perform relatively independently of one another. It is a specific program design technique, not a comprehensive design method. Thus it does not specify file or database design, input or output layout or the hardware on which the

application will run. It provides specification of program modules that are functionally independent.

It is a process-centred technique that transforms the structured analysis models into good software design models. Structured Design introduced a modeling tool called Structure Charts. They are used to illustrate software (program) structure to fulfill business requirements. Structure charts describe the interaction between independent module and the data passing between the modules. These module specifications can be passed to programmers prior to the writing of program code. In structure chart the whole application is divided into modules (set of program instructions) and modules are designed according to some principles of design. These principles are:

Modularity and Partitioning: Each system should consist of a hierarchy of modules. Lower level modules are generally smaller in scope and size compared to higher level modules. They serve to partition processes into separate functions.

Coupling: Modules should be loosely coupled. It means that modules should have little dependence on other modules in a system.

Cohesion: Modules should be highly cohesive. It means that modules should carry out a single processing function.

Span of Control: Modules should interact with and manage the functions of a limited number of lower level modules. It means that the number of called modules should be limited (in a calling module).

Size of Module: The number of instructions contained in a in a module should be limited so that module size is generally small.

Shared use of Functions: Functions should not be duplicated in separate modules may be shared. It means that functions can be written in a single module and it can be invoked by any other module when needed.

3.3 Prototype

A prototyping approach emphasizes the construction model of a system. Designing and building a scaled-down but functional version of a desired system is the process known as Prototyping. A prototype is a working system that is developed to test ideas and assumptions about the new system. It consists of working software that accepts input, performs calculations, produces printed or displayed Information or performs other meaningful activities. It is the first version or iteration of

an information system i.e. an origina1 model. Customer evaluates this model. This can be effectively done only if the data are real and the situations are live. Changes are expected as the system is used. This approach is useful when the requirements are not well defined. A prototype is usually a test model. It is an interactive process. It may begin with only new functions and be expanded to include others that are identified later. The steps of Prototyping process are depicted in Figure 1.4. They are:

Identify the user's known information requirements and features needed in the system.
Develop a working prototype.
Revise the prototype based on feedback received from customer
Repeat these steps as needed to achieve a satisfactory system.

Actual development of a working prototype is the responsibility of a systems analyst. The difference between a prototype model and an actual information system is that, a prototype will not include the error checking, input data validation, security and processing completeness of a finished application. It will not offer user help as in the final system.

But, sometimes, the prototype can evolve into the product to be built. The prototype can be easily developed with tools of fourth generation languages (4GL's) and with the help of Computer Aided Software Engineering (CASE) tools, Prototyping approach is a form of rapid application development (RAD).

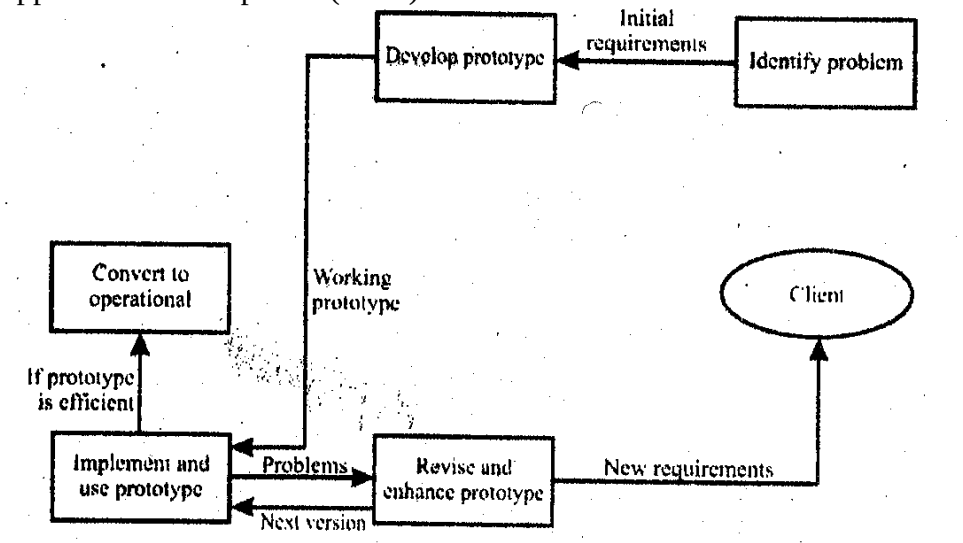


Figure 1.4: Prototype Approach

3.4 Joint Application Development

It is defined as a structured approach in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements. The important feature of JAD is joint requirements planning, which is a process whereby highly structured group meetings are conducted to analyze problems and define requirements.

The typical participants in a JAD are listed below:

JAD Session Leader: The JAD leader organizes and runs the JAD. This person is trained in group management and facilitation as well as system analysis. The JAD leader sets the agenda and sees that it is met. The JAD leader remains neutral on issues and does not contribute ideas or opinions but rather concentrates on keeping the group on the agenda, resolving conflicts and disagreements, and soliciting all ideas.

(1) **Users**: The key users of the system under consideration are vital participants in a JAD. They are the only ones who have a clear understanding of what it means to use the system on a daily basis.

(2) **Manager**: The role of managers during JAD is to approve project objectives, establish project priorities, approve schedules and costs and approve identified training needs and implementation plans.

(3) **Sponsors**: A JAD must be sponsored by someone at a relatively high level in the company i.e. the person from top management. If the sponsor attends any --session, it is usually at the very beginning or at the end.

(4) **Systems Analysts**: Members of the systems analysis team attend the JAD session although their actual participation may be limited. Analysts are there to learn from customers and managers, but not to run or dominate the process.

(5) **Scribe**: The scribe takes down the notes during the JAD sessions. This is usually done on a personal computer or a laptop. Notes may be taken using a word processor. Diagrams may directly be entered into a CASE tool.

(6) **IS staff** like systems analysts, other IS staff such as programmers, database analysts, IS planners and data centre personnel may attend to learn from the discussions and possibly to contribute their ideas on the technical feasibility of proposed ideas or on technical limitations of current systems.

The following are the various benefits of Joint Application Development:

actively involves users and management in project development,
reduces the amount of time required to develop a system, and
incorporates prototyping as a means for confirming requirements and obtaining design approvals.

4.0 CONCLUSION

You have learned about the different approaches; model drives, accelerated and JAD approach, for developing information systems.

5.0 SUMMARY

What you have learned in this unit concerns the various approaches for developing information systems.

SELF ASSESSMENT EXERCISE

1. Write briefly on the structure analysis and design approach.
2. Describe the joint application development.

6.0 TUTOR-MARKED ASSIGNMENT

Enumerate the steps involved in the prototyping process.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.rspa.com

UNIT 5 SYSTEMS ANALYST – A PROFESSION

CONTENTS

1.0 INTRODUCTION

This unit describes why business requires systems analysts. It defines system users and describes analysts in various functional areas.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

explain why business need systems analysts
define system users
describe analysts in various functional areas.

3.0 MAIN CONTENT

3.1 Why do Businesses Need Systems Analysts

A computerized system enables an organization to provide accurate information and respond faster to the queries, events etc. If a business needs computerized information system, a Systems Analyst is required for analysis and design of that system. Information systems evolved from the need to improve the use of computer resources for the information processing needs of business application. Customer defines the business problems to be solved by the computer. Project managers, Analysts, Programmers and Customers apply information technology to build information systems that solve those problems. Information technology offers the opportunity to collect and store enormous volume of data, process business transactions with great speed and accuracy and provide timely and relevant information for taking correct decision by management. This potential could not be realized without the help of a systems analyst since business users may not fully understand the capabilities and limitations of modern information technology. Similarly, computer programmers and information technologists do not fully understand the business applications they are trying to computerize or support. A communication gap has always existed between those who need computer based business solutions and those who understand information technology. Systems analyst bridges this gap.

3.2 Users

System users are defined as the people who use information systems or who are affected by the information system on a regular basis i.e. capturing, validating, entering, and responding to, storing and exchanging data and information apart from others. A common synonym is client. System users are concerned with business requirements. There are two main classes of system users and they are discussed below.

Internal Users are employees of the business for which an information system is built Examples are clerical and service staff, technical and professional staff, supervisors, middle level managers and executive managers. Remote and mobile users are the new class of internal users. They are geographically separated from the business. Examples of mobile users are sales and service representatives. An example of remote user is the person who is associated with telecommunication i.e. working from home. The person can be connected to the company's information system through modern communications technology.

External Users: Modern information systems are now reaching beyond the boundaries of the traditional business to include customers and other businesses as system users. In business-to-business information systems, each business becomes an external user of the other business's information systems. For example, in the case of direct purchasing of product through the Internet, customer becomes an external user of the retailer's order processing information systems. Another example is that if a business connects their purchasing systems directly to the order processing systems of their suppliers then both become external users to each other.

3.3 Analysts in Various Functional Areas

Today the systems analyst's job presents a fascinating and exciting challenge. It offers high management visibility and opportunities for important decision-making and creativity that may affect an entire organization.

3.3.1 Systems Analyst in Traditional Business

In the traditional business, information services are centralized for the entire organization or for a specific location. In this organization, the staff of information services (refer to Figure 2.1) report directly to the chief executive officer (CEO). The highest-ranking officer is sometimes called a chief information officer (CIO) and the rest of information services are organized according to the following functions or areas:

System Development: functional business, systems analysts and programmers are organized into permanent teams that support the information systems and applications for specific business function. System development unit includes a *centre for excellence*, which is a group of experts (experienced systems analysts, system designers, and system builders) who establish and enforce methods, tools, techniques and quality for all system development projects.

Data Administration: Data and other Information Resources of the organization are managed. This includes databases that are used by system developers to support applications. Systems analysts who are experts in data analysis can work here. These analysts are known as *Data Analysts.* They analyse database requirements, design and construct (sometimes) the corresponding databases.

Telecommunications: Here, computer networks that play a critical role in the success of any business are designed, implemented and managed. Here, Network *analysts* perform many of the tasks as applied to designing local and wide area networks that will ultimately be used by systems and applications.

End-user Computing: The growing base of personal computers and local area networks in the end user community are supported. This provides installation services, training and helps desk services. Analyst also provides standards and consulting to end users that develop their own systems with PC power tools such as spreadsheets and PC database management systems. In this centre, analysts may work as *End-user computing consultants.*

Computer Operations: All of the shared computers including mainframes, minicomputers and other computers are put to operation and the same is coordinated. Systems Analysts may work as *Capacity Analysts* in this area.

Every analyst should know the management structure of a traditional information services organization.

3.3.2 Systems Analyst in Modern Business

Many medium-to-large information services units for the modem business have reorganized to be decentralized with a focus on empowerment and dynamic teams. In modem business, systems analyst may be reassigned to different projects at time to time. During the project, the systems analyst and other team members are directly accountable to the business unit for which the system is being developed. In this type of organization, the information services try to get closer to users and management to improve services and value. Today's analysts should also know about a modem\information services organization.

In modem business, two new trends are used for software development: outsourcing and consulting. *Outsourcing* is the act of contracting an outside vendor to assume responsibility for one or more IT functions or

services. *Consulting* is the act of contracting with an outside vendor to assume responsibility for or participate in one or more IT projects.

SELF ASSESSMENT EXERCISE

1. What is a system user?
2. Describe the systems analyst in modern business

4.0 CONCLUSION

The need for systems analysts in business is explained in this unit. This unit also describes analysts in various functional areas.

5.0 SUMMARY

This unit focuses on the systems analysts and varies functional areas of a systems analyst.

6.0 TUTOR-MARKED ASSIGNMENT

Why do businesses require system analysts?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.freefore.ssays.com

http://www.rspa.com

UNIT 6 SYSTEMS ANALYST – ROLER DUTIES AND QUALIFICATIONS

CONTENTS

1.0 INTRODUCTION

In this unit, you will study the role and duties of a System Analyst. You will also learn about the qualification and skills required of a System Analyst.

2.0 OBJECTIVES

After going through this unit, you should be able to:

know the role of a system analyst in a learn for the benefit of the organization
understands different analytical skills that are important to systems development including problem identification, problem solving and systems thinking
know the need for basic technical skills even when systems are developed using rapid prototyping and code generators.

3.0 MAIN CONTENT

3.1 Roles of a System Analyst

The success of an infom1ation system development is based on the role of Systems analyst. Among several roles, some important roles are described below:

Change Agent: The analyst may be viewed as an agent of change. A candidate system is designed to introduce change and reorientation in how the user organization handles infom1ation or makes decisions. Then, it is important that the user accepts change. For user acceptance, analysts prefer user participations during design and implementation. Analyst carefully plans, monitors and implements change into the user domain because people inherently resist changes. In the role of a change agent, Systems Analyst may use different approaches to introduce changes to the user organization.

Investigator and Monitor: A systems analyst may, investigate the existing system to find the reasons for it's failure. The role of an investigator is to extract the problems from existing systems and create information structures that uncover previously unknown trends that may have a direct impact on organization. The role of a Monitor is to

undertake and successfully complete a project. In this role, analysts must monitor programs in relation to time, cost and quality.

Architect: The analyst's role as an architect is liaison between the user's logical design requirements and the detailed physical system design. As architect the analyst also creates a detailed physical design of candidate systems. A systems analyst makes the design of information system architecture on the basis of end user requirements. This design becomes the blue print for the programmers.

Psychologist: In system development, systems are built around people. The Analyst plays the role of psychologist in the way s/he reaches people, interprets their thoughts, assesses their behaviour and draws conclusions from these interactions. Psychologist plays major role during the phase of fact finding.

Motivator: System acceptance is achieved through user participation in its development, effective user training and proper motivation to use the system. The analyst's role as a motivator becomes obvious during the first few weeks after implementation and during times when turnover results in new people being trained to work with the candidate system.

Intermediary: In implementing a candidate system, the analyst tries to appease all parties involved. Diplomacy in dealing with people can improve acceptance of the system. The analyst's goal is to have the support of all the users. S/he represents their thinking and tries to achieve their goals through computerization.
These multiple roles require analysts to be orderly, approach a problem in a logical way, and pay attention to details. They prefer to concentrate on objective data, seek the best method, and be highly prescriptive. They appear to be cool and studious.

They focus on method and plan, point out details, are good at model building, perform best in structured situations, and seek stability and order.

3.2 Duties of a Systems Analysts

The duty of a systems analyst is to coordinate the efforts of all groups to effectively develop and operate computer based information systems. The duties of a systems analyst are following:

Defining Requirements: The most important and difficult duty of an analyst is to understand the user's requirements. Several fact-finding techniques are used like interview, questionnaire, and observation, etc.

Prioritising Requirements by Consensus: There is a need to set priority among the requirements of various users. This can be achieved by having a common meeting with all the users and arriving at a consensus. This duty of systems analyst requires good interpersonal relations and diplomacy. S/he must be able to convince all the users about the priority of requirements.

Analysis and Evaluation: A systems and analyses the working of the current information system in the organization and finds out the extent to which they meet user's needs. On the basis of facts and opinions, systems analyst finds the best characteristics of the new or modified system which will meet the user's stated information needs.

Solving Problems: Systems analyst is basically a problem solver. An analyst must study the problem in depth and suggest alternate solutions to management. Problem solving approach usually incorporates the following general steps:

Identify the problem
Analyse and understand the problem
Identify alternative solutions and select the best solution.

Drawing up Functional Specifications: The key duty of systems analyst is to obtain the functional specifications of the system to be designed. The specification must be non-technical so that users and managers understand it. The specification must be precise and detailed so that it can be used by system implementers.

Designing Systems: Once the specifications are accepted, the analyst designs the system. The design must be understandable to the system implementer. The design must be modular to accommodate changes easily. An analyst must know the latest design tools to assist implementer in his task. An Analyst must also create a system test plan.

Evaluating Systems: An analyst must critically evaluate a system after it has been in use for a reasonable period of time. The time at which evaluation is to be done, how it is to be done and how user's comments are to be gathered and used, must be decided by the analyst.

3.3 Qualifications of a Systems Analyst

A systems analyst must fulfill the following requirements:

Working knowledge of information technology
Computer programming experience and expertise
General business knowledge .Problem solving skills

Communication skills .Interpersonal skills
Flexibility and adaptability
Thorough knowledge of analysis and design methodologies.

In summary, the skills that are required may be classified into the following:

Analytical skills
Technical skills
Management skills
Interpersonal skills.

3.3.1 Analytical Skills

As the designation of person is Systems Analyst, possession of analytical skills is very important. Analytical skills can be classified into the following sets:

System study
Organizational knowledge
Problem identification
Problem analysis and problem solving.

System Study: The first important skill of systems analyst is to know about system. It means that Systems Analyst should be able to identify work assignment as a system. It involves identification of each of the system's characteristics such as inputs, outputs, processes etc. Information systems can be seen as subsystems in larger organizational systems, taking input and returning output to their organizational environments.

Data flow diagram clearly illustrates inputs, outputs, system boundaries, the environment, subsystems and inter-relationship. Purpose and constraints are much more difficult to illustrate and must therefore be documented using other notations. In total, all elements of logical system description must address all characteristics of a system.

Organizational Knowledge: Whether a person is an in-house (in traditional organization) or contract software developer (in modern organization), sine must understand how organization works. In addition s/he must understand the functions and procedures of the particular organization (or enterprise) s/he is working for. Selected areas of organizational knowledge for a systems analyst are given below:

(1) Howworkofficiallygetsdoneinaparticularorganization: In this area, knowledge about the following is required:

Terminology, abbreviations and acronyms
Policies
Standards and procedures
Formal organization structure
Job description.

(2) Understandingtheorganization'sinternalpolitics: In this area, knowledge is required about the following:

Influence and inclinations of key personnel
Finding the experts in different concerned subject areas
Critical events in the organization's history
Informal organization structure
Coalition membership and power structures.

(3) Understanding the organization's competitive and regulatory environment: In this area, knowledge is required about the following:

Government regulations
Competitors from domestic and international fronts
Products, services and markets
Role of technology.

(4) Understanding the organization's strategies and tactics: In this area, the requisite knowledge is given below:

Short as well as long term strategy and plans
Values and mission

Problem Identification: A problem can be defined as the difference between an existing situation and a desired situation. The process of identifying problem is the process of defining differences. So, problem solving is the process of finding a way to reduce differences. A manager defines differences by comparing the current situation to the output of a model that predicts what the output should be. In order to identify problems that need to be solved, the systems analyst must develop a repertoire of models to define the differences between what is present and what ought to be present.

Problem Analysis and Problem Solving: Once a problem has been identified, systems analyst must analyse the problem and determine how to solve it. Analysis entails more about the problem. Systems analyst learns through experience, with guidance from proven methods, the process of obtaining information from concerned people as well as from

organizational files and documents. As s/he seeks out additional information, s/he also begins to formulate alternative solutions to the problem. The next step is that the alternatives are compared and typically one is chosen as best solution. Once the analyst, users and management agree on the general suitability of a solution (feasibility), they devise a plan for implementing it.

Herbert Simon has first proposed this approach. According to her/him, this approach has four phases namely intelligence, design, choice and implementation. This approach is similar to system development life cycle.

Intelligence: During this phase, all information relevant to the problem is collected.
Design: During this phase, alternatives are formulated.
Choice: During this phase, the best alternative solution is chosen.
Implementation: During this phase, the solution is put into practice.

3.3.2 Technical Skills

Many aspects of the job of systems analyst are technically oriented. In order to develop computer based information systems, systems analyst must understand information technologies, their potentials and their limitations. A systems analyst needs technical skills not only to perform tasks assigned to him/her but also to communicate with the other people with whom s/he works in systems development. The technical knowledge of a Systems Analyst must be updated from time to time.

In general, a Systems Analyst should be as familiar as possible with such families of technologies such as:

Microcomputers, workstations, minicomputers, and mainframe computers, Programming languages,
Operating systems, both for PC's and networks,
Database and File management systems,
Data communication standards and software for local and wide area networks, System development tools and environments (such as forms & report generators and graphical user interface design tools), and
Decision support systems and data analysis tools.

S/he should know all of the above as well as modern methods and techniques for describing, modeling and building systems.

3.3.3 Management Skills

When a systems analyst is asked to lead a project team then management skills are required. Systems analyst needs to know the process of managing his/her own work and how to use organizational resources in the most productive ways possible. Self-management is important skill for an analyst. There are four categories of management skills:

Resource management
Project management
Risk management
Change management

Resource Management: A systems analyst must know how to get the most out of a wide range of resources i.e. system documentation, information technology and money. A team leader must learn how to best utilize the particular talents of other team members. S/he must also be able to delegate responsibility, empower people to do the tasks they have been assigned.

Resource management includes the following capabilities:

Predicting resource usage (budgeting)
Tracking and accounting for resource consumption
Learning how to use resources effectively
Evaluating the quality of resources used
Securing resources from abusive use
Relinquishing resources when no longer needed and releasing the resources when they can no longer be useful.

Project Management: A *project* is defined as a sequence of unique, complex and connected activities having one goal or purpose and that must be completed by a specific time, within budgets and according to specifications.

Project management is defined as the process of scoping, planning, staffing, organizing, directing and controlling the development of acceptable system at minimum cost within a specified time frame. In the role of project manager, s/he first needs to decompose a project in to several independent tasks. The next step is to determine how the tasks are related to each other and who will be responsible for each task.

Risk Management: A risk is any unfavourable event or circumstance that can occur while a project is underway. If a risk comes true, it can hamper the successful and timely completion of a project. Therefore, it is necessary to anticipate and identify different risks, a project is susceptible to, so that contingency plans can be prepared in advance to control the effects of each risk. Once, risk to the project has been identified, project manager must be able to minimize the likelihood that those risks will actually occur. It also includes knowing where to place resources (such as people) where they can do the best and prioritising activities to achieve better productivity.

Change Management: Introducing a new or improved information system into an organization is a change process. In general people do not like change and tend to resist it. Therefore, any change in the way people perform their duties in an organization must be carefully managed. Change management is a very important skill for systems analyst. The systems analyst must know how to get people to make a smooth transition from one information system to another, giving up their old ways of doing things and accepting new ways. Change management also includes the ability to deal with technical issues related to change, such as obsolescence and reusability.

3.3.4 Interpersonal Skills

Systems analyst works extensively with staff in key positions in an organization. So, interpersonal skills are necessary for success of him/her. These skills can be classified as:

Communication skills
Working alone as well as in a team
Facilitating groups
Managing expectations.

Communication Skills: A Systems analyst should be able to communicate clearly and effectively with others. S/he must establish a good relationship with clients early in the project and maintain it throughout the project. Communication takes many forms from written to verbal to visual. The analyst must be able to master as many forms of communication as possible. Interpersonal communication subjects are:

Business speaking .Business writing
Interviewing
Listening
Technical discussion
Technical writing.

Working alone as well as in a team: A Systems analysts must be able to organize and manage his/her own schedule, commitments and deadlines because many people in the organization will depend on his/her individual performance, but systems analyst must work with the team towards achieving project goals. To work together effectively and to ensure the quality of the product, the team must establish standards of cooperation and coordination that guide their work. There are 12 characteristics of a high performance team that influence team work:

Shared and elevated vision
Sense of team identity: Result-driven structure
Competent team members
Commitment to the team
Mutual trust
Interdependency among team members
Effective communication
Sense of autonomy
Sense of empowerment
Small team size.

Facilitating Groups: This skill is required when systems analyst works in Joint application development approach. In this approach systems analyst works with group during system development. Analysts use JAD sessions to gather systems requirements and to conduct design reviews. Systems analyst can be asked to work as a facilitator. Facilitation necessarily involves a certain amount of neutrality on the part of the facilitator. The facilitator must guide the group without being a part of the group and must work to keep the effort on track by helping the group resolve differences. Guidelines for a facilitator are given below:

Purpose should be made clear
Make sure that the group understands what is expected of them and of you
Use physical movement to focus on yourself or on the group
Reward group member participation with thanks and respect
Ask questions instead of making statement
Wait patiently for answers
Be a good listener
Encourage group members to feel ownership of the group's goal and of their attempts to reach those goals.

Managing Expectations: System development is a change process, and members of any organization greet any organizational change with anticipation and uncertainty. Organizational members will have certain ideas about what new information system will be able to do for them.

Ginzberg found that successfully managing user expectations is related to successful systems implementation: The systems analyst needs to understand the technology. S/he must understand the work flows that the technology will support and how the new system will affect them. The important ability of systems analyst is to communicate a realistic picture of the new system and what it will do for users and managers. Managing expectations begins with the development of the business case for the system and extends all the way through training people to use the finished system.

The relationship between a systems analyst's skills and systems development life cycle are depicted in figure 2.1. Systems analyst works extensively with staff in key positions in an organization. So, interpersonal skills are necessary for success of him/her. These skills can be classified as:

Communication skills
Working alone as well as in a team
Facilitating groups
Managing expectations.

Communication Skills: A Systems analyst should be able to communicate clearly and effectively with others. S/he must establish a good relationship with clients early in the project and maintain it throughout the project. Communication takes many forms from written to verbal to visual. The analyst must be able to master as many forms of communication as possible. Interpersonal communication subjects are:

Business speaking .Business writing
Interviewing
Listening
Technical discussion
Technical writing.

Working alone as well as in a team: A Systems analysts must be able to organize and manage his/her own schedule, commitments and deadlines because many people in the organization will depend on his/her individual performance, but systems analyst must work with the team towards achieving project goals. To work together effectively and to ensure the quality of the product, the team must establish standards of cooperation and coordination that guide their work. There are 12 characteristics of a high performance team that influence team work:

Shared and elevated vision
Sense of team identity: Result-driven structure
Competent team members

Commitment to the team
Mutual trust
Interdependency among team members
Effective communication
Sense of autonomy
Sense of empowerment
Small team size.

Facilitating Groups: This skill is required when systems analyst works in Joint application development approach. In this approach systems analyst works with group during system development. Analysts use JAD sessions to gather systems requirements and to conduct design reviews. Systems analyst can be asked to work as a facilitator. Facilitation necessarily involves a certain amount of neutrality on the part of the facilitator. The facilitator must guide the group without being a part of the group and must work to keep the effort on track by helping the group resolve differences. Guidelines for a facilitator are given below:

Purpose should be made clear
Make sure that the group understands what is expected of them and of you
Use physical movement to focus on yourself or on the group
Reward group member participation with thanks and respect
Ask questions instead of making statement
Wait patiently for answers
Be a good listener
Encourage group members to feel ownership of the group's goal and of their attempts to reach those goals.

Managing Expectations: System development is a change process, and members of any organization greet any organizational change with anticipation and uncertainty. Organizational members will have certain ideas about what new information system will be able to do for them. Ginzberg found that successfully managing user expectations is related to successful systems implementation: The systems analyst needs to understand the technology. S/he must understand the work flows that the technology will support and how the new system will affect them. The important ability of systems analyst is to communicate a realistic picture of the new system and what it will do for users and managers. Managing expectations begins with the development of the business case for the system and extends all the way through training people to use the finished system.

The relationship between a systems analyst's skills and systems development life cycle are depicted in figure 2.1.

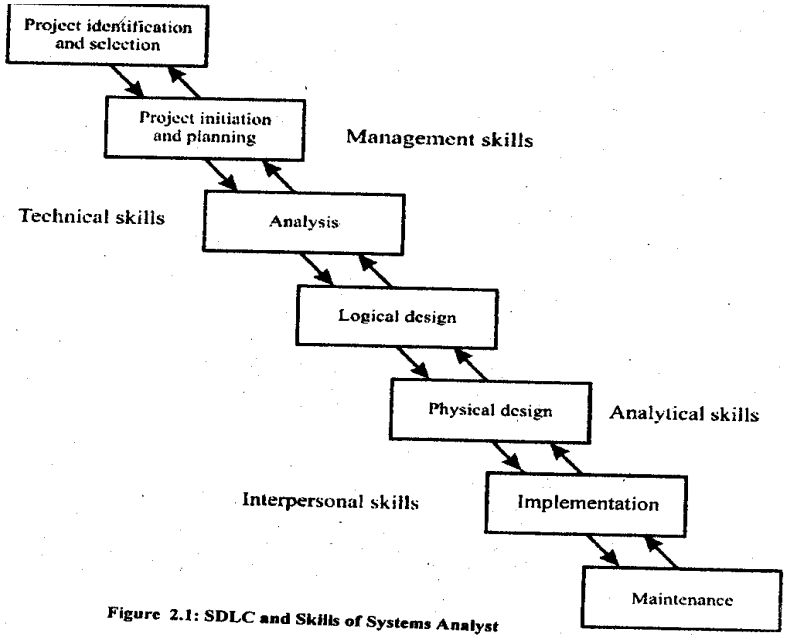


Figure 2.1: SDLC and Skills of Systems Analyst

SELF ASSESSMENT EXERCISE

1. List at least six attributes of a System Analyst?
2. Why should a System Analyst be able to communicate well?

4.0 CONCLUSION

You have studied the role and duties of a System Analyst. You have equally learned about the qualification and skills required of a system Analyst.

5.0 SUMMARY

What you have learned in this unit focuses on the role, duties and qualification of a System Analyst.

6.0 TUTOR-MARKED ASSIGNMENT

Describe the duties of a System Analyst?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.freeforessays.com
http://www.rspa.com

UNIT 7 PROCESS OF SYSTEM DEVELOPMENT

CONTENTS

1.0 INTRODUCTION

In this unit you will learn about the systems development life cycle (SDLC). You will also learn about the phases of SDLC.

2.0 OBJECTIVES

By the end of this unit, you should be able to:

describe the information systems development life cycle
explain the phases of SDLC
list the three types of maintenance in the SDLC.

3.0 MAIN CONTENT

3.1 Systems Development Life Cycle

Most organizations find it beneficial to use a set of steps, called a systems development methodology, to develop and support their information system. Like many processes, the development of information system often follows a life cycle. The system development life cycle (SDLC) is a common methodology for system development in many organizations, featuring various phases that mark the progress of the system analysis and design effort.

Although any life cycle appears at first glance to be a sequentially ordered set of phases but actually it is not. The specific steps and their sequence are meant to be, adapted as required for a project, consistent with management approach. For example, in any given SDLC phase, the project can return to an earlier phase, if necessary. If a commercial product does not perform well just after its introduction, it may be temporarily removed from the market and improved before being re-introduced. In the system development life cycle, it is also possible to complete some activities in one phase in parallel with some other activities of another phase. Sometimes, life cycle is iterative; that is, phases are repeated as required until a satisfactory and acceptable system is found. Such an iterative approach is special characteristic of rapid application development methods, such as prototyping. Some people consider life cycle to be spiral, in which we constantly cycle through the phases at different levels of detail. The life cycle can also be thought of a circular process in which the end of the useful life of one system leads to the beginning of another project that will develop a new version or replace an existing system altogether. However, the system development life cycle used in an organization is an orderly set of activities conducted and planned for each development project? The skills of a system analyst are required to be applied to the entire life cycle.

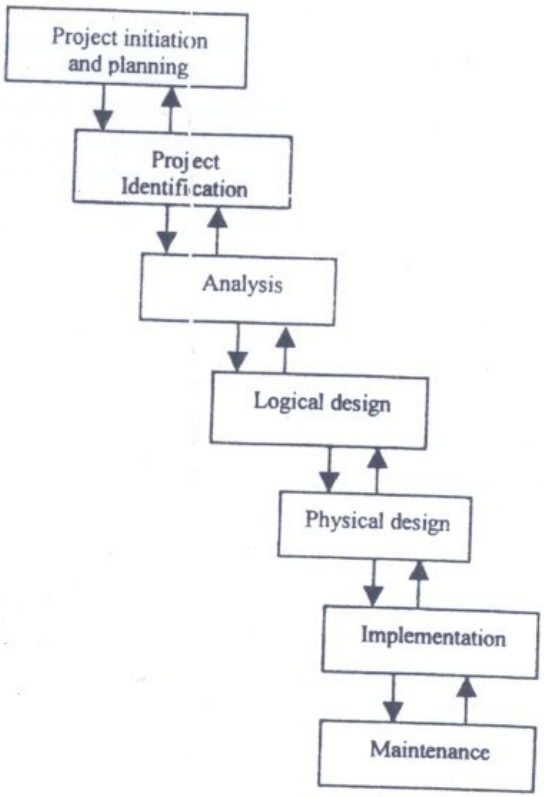


Figure 3.3: Phases of System Development Life Cycle

Every custom software producer Will nave its own specific detailed life or system development methodology. Even if a particular methodology does not look like cycle, you will discover that many of SDLC steps are performed, SDLC techniques and tools are used.

In order to make this unit generic, we follow a rather general life cycle model, as described in figure 3.1.

This model resembles a staircase with arrows connecting each step to the step before and to the step after it. This representation of the system development life cycle (SDLC) is sometimes referred to as the "waterfall model". We use this SDLC as one example of methodology but more as a way to arrange the steps of systems analysis and design. Eachpl1ase has specific outputs and deliverables that feed important information to other phases. At the end of each phase, system development project reaches a milestone and, as deliverables are produced, parties outside the project team often review them.

3.2 Phases of SDLC

SDL consists of mainly seven steps. These are:

1. Project Identification and Selection

2. Project Initiation and Planning
3. Analysis
4. Logical Design
5. Physical Design
6. Implementation
7. Testing.

3.2.1 Project Identification and Selection

The first phase in the SDLC is called project identification and selection. In this phase, the user identifies the need for a new or improved system. In large organizations, this identification may be part of a systems planning process.

Information requirements of the organization as a whole are examined, and projects to meet these requirements arc proactively identified. The organization's information system requirements may result from requests to deal with problem in current system's procedures, from the desire to perform additional tasks, or from the realization that information technology could be used to capitalize on an existing opportunity. These needs can then be prioritised and translated into a plan for the Information System department including a schedule for developing new major systems. In smaller organizations, determination of which systems to develop may be affected by user request submitted as the need for new or enhanced systems arises as well as from a formal information p1anning process. In either case, during project identification and selection, an organization determines whether or not resources should be devoted to the development or enhancement of each information system under consideration. The outcome of the project identification and selection process is a determination of which systems development projects should be undertaken by the organization at least in terns of an initial study.

3.2.2 Project Initiation and Planning

The second phase is project initiation and planning. The problems that are identified should be investigated and a decision to implement the information system or not for the organization should be taken. A critical step at this point determining the scope of the proposed system. The project leader and initial team of system analysts also produce a specific plan for the proposed project, which the team will follow using the remaining SDLC steps. Now, this baseline project plan customizes the standardized SDLC and specifies the time and resources needed for its execution.

The formal definition of a project is based on the likelihood that the organization's information system department is able to develop a system that will solve the problem or use the opportunity and determine whether the costs of developing the system outweigh the benefits it could provide. The final presentation with the subsequent project phase is usually made by the project leader and other team members to someone in management or to a special management committee with the job of deciding which projects the organization will undertake.

3.2.3 Analysis

Analysis is the next phase. During this phase, the analysis has several sub-phases. The first is requirements determination. In this sub-phase, analysts work with users to determine the expectations of users from the proposed system. This sub-phase usually involves a careful study of current systems, manual or computerized that might be replaced or enhanced as part of this project. Next, the requirements are studied and structured in accordance with their inter-relationships and eliminate any redundancies. Third, alternative initial design is generated to match the requirements. Then, these alternatives are compared to determine which alternative best meets the requirement in terms of cost and labour to commit to development process.

In this phase, feasibility study of the proposed system is also performed. Various types of feasibilities are:

Technical feasibility
Economic feasibility
Behavioural feasibility
Operational feasibility
Legal feasibility
Time feasibility.

If the proposed system is not feasible to develop, it is rejected at this very step.

The output of the analysis phase is a description of (but not are detailed design for) the alternative solution recommended by1he analysis team. Once, the recommendation is accepted by those with funding authority, you can begin to make plans to acquire any hardware and system software necessary to build or operate the system proposed.

System Design: After analysis phase is complete, design of the system begins. The design consists of logical and physical design of the system. The fourth and fifth phases are devoted to design of the new and enhanced system. During design, you and the other analysts convert the

description of the recommended alternative solution into logical and then physical system specifications. You must design all aspects of the system from input and output screens to reports, databases, and computer processes. Design occurs in two phases, viz., logical design and physical design.

3.2.4 Logical Design

Logical design is not tied to any specific hardware and systems software platform. Theoretically, the system could be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended. Logical design concentrates on the business aspects of the system.

3.2.5 Physical Design

In physical design, the logical design is turned into physical or technical specifications. For example, you must convert diagrams that map the origin, flow, processing or data in a system into a structured systems design that can then be broken down into smaller and smaller units known as modules for conversion to instruction written in a programming language. You design various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output. During the physical design, the analyst team decides the programming language in which the computer instructions will be written in, which database system and file structure will be used for the data, the platform that will be used and the network environment under which the system will be run. These decisions finalize the hardware and software plans initiated at the end of the analysis phase. Now, proceedings can be made with respect to acquisition of any new technology not already present in the organization. The final product of the design phase is the physical system specification in a form ready to be turned over to programmers and other system builders for construction. The physical system specifications are turned over to programmers as the first part of the implementation phase.

3.2.6 Implementation

During implementation, you turn system specification into working system that is tested and put into use. Implementation includes **coding, testing and installation**. During coding, programmers write programs that make up the system. During testing, programmers and analysts test the individual programs and the entire system in order to find and correct errors. During installation, the new system becomes a part of the daily activities of the organization. Application is installed or loaded, on

existing or new hardware and users are introduced to new system and trained. The analysts begin planning for testing and installation as early as the project initiation and planning phase, since testing and installation require extensive analysis in order to develop the right approach.

Installation of the system can be done in the following three ways:

Direct conversion: In this type of conversion, the software is directly installed at user's site.

Parallel conversion: In this type of conversion, both the old and new systems are run in parallel for some time. After monitoring the new system for a reasonable period of time and if it is performing well, then, the new system is implemented replacing the old one.

Phased conversion: In this type of conversion, the system is installed module by module.

Implementation activities also include initial user support such as the finalization of *documentation, training programs, and ongoing user assistance.* Note that documentation and training programs are finalized during implementation. Documentation is produced throughout the lifecycle. Implementation can continue for as long as the system exists since ongoing user support is also part of implementation. Despite the cost efforts of analysts, managers, and programmers, however, installation is not always a simple process. Many well-designed systems can fail if implementation is not well managed. The management of implementation is usually done by the project team.

3.2.7 Maintenance

The final phase is maintenance. When a system is operating in an organization, users sometimes find problems with how it works and often think of better ways to perform its functions. Also, the organization's requirements with respect to the system change with time. During maintenance, programmers make the changes that users ask for and modify the system to reflect and support changing business conditions. These changes are necessary to keep the system running and useful. Maintenance is not separate phase but a repetition of the other lifecycle phases required to study and implement the needed changes. Thus, maintenance is an overlay to the life cycle rather than a separate phase. The amount of time and effort devoted to maintenance depends a great deal on the performance of the previous phase of life cycle. There comes a time, however, when an information system is no longer performing as desired, when maintenance cost becomes prohibitive, or when the organization's needs has changed substantially. Such problems

are an indication that it is the time to begin designing the system's replacement, therefore, completing the loop and starting the life cycle over again. Often, the distinction between the major maintenance and new development is not clear, which is another reason why maintenance often resembles the lifecycle itself.

Maintenance is of three types:

Corrective maintenance: In this type, the errors that creep into the system are removed. Hence the name *corrective maintenance.*

Adaptive maintenance: It is done to adapt with the changing external factors. For example, if the government rules change regarding the Dearness Allowance from 52% to 58%, then the changes have to be made in the Information System to adapt with the changing scenario.

Perfective maintenance: This is done to satisfy the users' requirements to make the system more and more perfect.

The SDLC is a highly linked set of phases where output of one phase serves as input to the subsequent phase. Throughout the systems development life cycle, the systems development project needs to be carefully planned and managed. Therefore, the larger the project, the greater is the need for project management.

SELF ASSESSMENT EXERCISE

1. Enumerate five types of feasibilities in the analysis phase
2 Describe three ways of installing a system

4.0 CONCLUSION

What you have learned in this unit includes the system development life cycle and the phases of SDLC.

5.0 SUMMARY

This unit focuses on the system development life cycle SDLC.

6.0 TUTOR-MARKED ASSIGNMENT

List the seven steps involved in the system development life cycle?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

By Jeffrey A., Hoffer, Joey F. George, Joseph S. Valacich; (2002). *Modern Systems Analysis and Design*; Pearson Education; (Third Edition).

Online Resources

http://www.rspa.com
http://www.leee.org

UNIT 8 SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

CONTENTS

1.0 INTRODUCTION

This unit describes the products of the system development life cycle phases. It equally introduces the different approaches to development. Finally, it presents a case study of computerizing the library of a college.

2.0 OBJECTIVES

After going through this unit, you should be able to:

describe the production of SDLC phases
explain Prototyping
describe the Joint application design
explain the participating design
describe a typical case study.

3.0 MAIN CONTENT

3.1 Products of SDLC Phases

Project Identification and Selection: Priorities for systems and project, architecture for data, networks, hardware and Information System Management are the result of the associated system.
Project Initiation and Planning: Detailed work plan for project, specification of system scope and high level system requirements, assignment of team members and other resources.

Analysis: Description of current system, need to enhance or replace current system, explanation of alternative systems and justification of alternatives. *Logical Design:* Functional and detailed specification of all system elements (data, process, input and output).

Physical Design: Technical, detailed specifications of all system elements, i.e., .programs, files, network, system software, etc. and acquisition plan for new technology.

Implementation: Code, documentation, training programs and support capabilities.

Maintenance: New version of software with associated updates of documents, training and support. life cycle (SDLC) is a common methodology for system development in many organizations, featuring various phases that mark the progress of the system analysis and design effort.

3.2 Approaches to Development

In the continuing effort to improve the systems analysis and design process, several approaches have been developed. Attempts to make system development less of an art and more of a science usually referred to as engineering techniques, are applied to system development. We will discuss prototyping, followed by introduction to joint application design and participatory design.

3.2.1 Prototyping

Designing and building a scaled down but fundamental version of a desired system is known as prototyping. A prototype can be built with any computer language or development tool to simplify the process. A prototype can be developed with some fourth generation languages (4GLs) such as query, screen and report design tools of a data base management system (DBMS), and with tools called computer aided software engineering (CASE) tools.

Using prototyping as a development technique, the analyst works with user to determine the initial or basic requirements of the system. The analyst then builds a prototype. When the prototype is completed, the user works with it and tells the analyst what they like and do not like about it .The analyst uses this feedback to improve the prototype and take the new version back to the user. This process is iterated until the users are satisfied. Two key advantages of the prototyping technique are the large extent to which proto typing involves the user in analysis and design and its ability to capture requirements in concrete rather than verbal or abstract form. In addition to being used stand-alone, prototyping can also be used to augment the SDLC.

Prototyping is a form of rapid application development or RAD. The fundamental principle of any RAD methodology is to delay producing detailed system design document until the user requirements are clear. The prototype serves as the working description of needs. RAD methodologies emphasize gaining user acceptance of the human system interface and developing core capabilities as quickly as possible.

3.2.2 Joint Application Design

In the late 1970s, systems development personnel at IBM developed a new process for collecting information system requirements and reviewing systems designs. The process is caused Joint Application Design (JAD). The basic idea behind JAD is to bring structure to the requirements determination phase of analysis and to the reviews that occur as part of design. Users; managers, and system developers are brought together for a series of intensive structured meetings run by a JAD session leader who maintains the structure and sticks to the agenda.

By gathering the people directly affected by an Information System in one room at the same time to work together to agree on system requirements and design details, time and organizational resources are better put to use. An added advantage is that, group members are more likely to develop a shared understanding of what the information system is supposed to do.

3.2.3 Participatory Design (PD)

Participatory Design (PD) represents a useful alternative approach to the SDLCPD emphasizes the role of the user much more than other techniques do. In some cases, PD may involve the entire user community in the development process. Each user has an equal share in determining system requirements and in approving system design. In other cases, an elected group of users control the process. These users represent the larger community. Under PD, systems analysts work for the users. The organization's management and outside consultants provide advice rather than control. PD is partly a result of the role of labour and management in the workplace where labour is more organized and is more intimately involved with technological changes.

3.3 Case Study

The problem is to computerize Library of XYZ College. In this library, all transactions are handled manually. Registers are maintained to record the details of the books, information about the members of the library and to manipulate the issue and return of books. The data entry and recovery procedures are all manual and this takes a lot of time and energy to browse through the pages of the register for locating the relevant information.

This current manual system of the library is very tough and time-consuming and chances of getting errors gets very high. This method is not trustworthy. This problem can be solved in the following steps:

Project Initiation and Planning

The specific services provided by our project will, of course, differ from other projects. Understanding the reasons behind the development of this project gives an appreciation of what our project does.

The main objective behind this project is:

To provide the user with an easy and fast interface.
To see that information handling is very easy and fast.
Easy updation and modification of data.

The basic aim of the project is to automate the basic functions of the library:

- To handle the Book Details.
- To record and handle the Member Details.
- To handle issue, return of books and to keep details about the books given for reading.

Is it feasible to automate the system? The three major areas to determine the feasibility of project are given below:

Technical Feasibility: The current level of technology can support the proposed system. The proposed software is enabled to meet all the objectives of the system and the output received would be more efficient. So, the project is technically feasible.

Economic Feasibility: The proposed system needs to get hardware and software installed. The short-term costs are overshadowed by the long-term gains. The management in question can invest in the system and is in condition to pay for the cost of system's study, cost of employee's time involved in the study and the cost of development of software. Thus, project is economically feasible.

Operational Feasibility: The current system faces a lot of problems which would be removed in the proposed system. The employees of the system will be free from the burden of the paper work and a lot of confusion. The employees are themselves interested in getting the manual system replaced by the automated one. The proposed system is user friendly. So, even a layman can use it. Thus, it is operationally feasible.

Design

Once it is found that the project development is feasible, Design has to be developed for the requirements listed in the analysis phase.

3.4 Data Dictionary

A Data Dictionary is a catalogue of all elements in a system. It consists of data about data.

It is a document that collects co-ordinates and confirms what specific data terms mean to different people in the team.

It is important for the following reasons:

- to manage the details,
- communicate meaning,
- document system features,

- facilitate analysis, and
- locate errors and omissions.

Consider a Library Information System. Our Data Dictionary record stores the following descriptions:

Book_Details

Stores information about books in the library. The table contains the following attributes:

Attributes	Stores
Book Id (Primary key)	ISBN Number
Name	Name of the book
Category	Category of the book
Subject	Subject of the book
Author	Author of the book
Price	Price of the book
Date	Date on which the book is added in the library
Edition	Edition of the book
Copies	Total no. of copies of the book present in the library

Book _Copy

Stores information about copies of a book available in the library.

Attributes	Stores
Book Id	ISBN Number
Book Idg	Indent no. of the book whose multiple copies are present

Book_State

Stores information regarding current status of the book.

Attributes	Stores
Book Id (primary key)	ISBN Number
Status	It gives information about current status of the book. Status of a book can be:
	I=> Book is issued.
	L=> Book is lost.
	P=> Book is present.
	M=> Book is lost by member.
	D=> Book is deleted.

Book_IR

It gives information about the state of the book which is issued to a member.

Attributes	Stores
TID (primary key)	Identification no. of issue/return transaction
Book ID	ISBN Number
MemID	Stores indent of member who issued the book
Mode	Mode can be:
	I=> Book is issued for home.
	R=> Book is issued for reading in library
State	State can be:
	O=> Book is with the member
	I=> Member returned the book
	2=> Member lost the book

Book_Irdetail

It gives information about issue/return and date/time of the issue/return of the book.

Attributes	Stores
TID	Identification number of the transaction of issue/return
Issue_Date	Stores the date on which the book is issued
Issue_Time	Stores the time the book is issued
Return_Date	Stores the date on which a member returns the book
Return_Time	Stores the time at which a member returns the book

Book_Delete

It stores the information about the books that are deleted and the date on which it was deleted.

Attributes	Stores
BookId	ISBN Number
Date	Stores the date on which the book was deleted.

Book_Renewed

It stores the information about the book, when it is bought back to the library and the date on which it was renewed.

Attributes	Stores

BookId	Stores the bookId of the book being deleted
Date	Stores the date on which the book was resumed

Member_Details

It stores the information about members of the library

Attributes	Stores
MemId	Stores the ID of the member
First	Stores the first name of the member
Middle	Stores the middle name of the member
Last	Stores the last name of the member
Sex	Stores the gender of the member
Address	Stores the address of the member
City	Stores the city of the member
State	Stores the state of the member
Country	Stores the country of the member
PIN	Stores the pin code of the member
Age	Stores the age of the member
Phone	Stores the phone number of the member
Issue_limit	Stores the maximum number of books that can be issued to the member
Date of Joining	Stores the date on which member enrolls in the library
Books Issued	Stores the number of books issued to the member

Member_State

Storing information about state of the member in the library

Attributes	Stores
MemID	Stores the ID of the member
State	It can have two values.
	P=> Member can access the library
	D=> Member is deleted fro library access

Member_Deleted

It stores information about the number deleted and the date on which the member was deleted.

Attributes	Stores
MemID	Stores identification no. of the member being deleted
Date	Stores date on which member was deleted

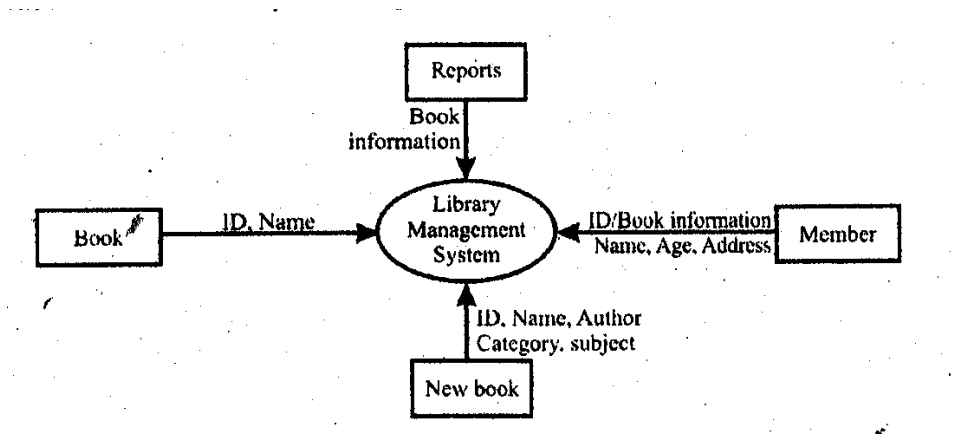


Figure 3:2: 0-level DFD of Library Information System

Input Design

The input design of this project is as follows. Points considered for the design of easy to fill out' form are given below which conforms to the design of the project:

Designing form with proper flow.
Logical grouping of information.
Labels holding suitable captions & textboxes to accept the data.
Usage of other tools, such as radio buttons, checkboxes, combo boxes, etc. also serve purpose for the better recording, processing, storing and retrieval of information.
The appearance of the form has been tried to be kept as attractive as possible to help in better and logical organization of details.
Since we know good screen design like good form design is an important instrument for steering the course of work, our design of input is guided by the following six objectives:

- Effectiveness .Accuracy
- Ease to use
- Consistency
- Simplicity
- Attractiveness.

Our screens show only that data which is necessary for the particular action being undertaken.

Screens are kept consistent by locating information in the same area each time a new screen is accessed.

We have made it easy to move from one screen to another through the use of icons, which channels the way to other screens apart from direct access to screens through the main menu.

Rather than jamming all data into one screen and cluttering up the screen, we have made use of multiple screens which add to the user appeal, thus are more productive and are prone to less errors.

3.5 Data Capture Information

Identification of data

The identifying data item in each transaction record is called a "KEY". Therefore, details of member is identified by the unique code, the details of the books through a unique book code. Similarly, the issue/return transactions through the transaction code.

Details of the Retrieval System

This in with reference to the stored data that can be quickly retrieved from the system files. This is done when we perform search on a particular criterion to draw the records or details of the search parameters.

Output Design

Users generally merit the system by its output. Thus, in order to create the most useful output, system analyst works closely with the user through the interactive process, until the result is considered to be satisfactory.

The objectives of the output design are:

Serve the intended purpose.
Output should satisfy the user.
Assured output where it is needed.
Output on time.
Choose appropriate output methods.

- Reports
- Messages (on screen)
- Document on help

Depending on the circumstances and the contents, the output may be displayed or printed. Output contents may originate from these sources:

Retrieval from data stores.
Transmission from a process or system activity.
Directly From input source.

Keeping the above points in mind, we have taken best care to present our information with the most clear and readable output. Our details are convincing enough to make the decisions fast and accurate.

Our reports represent one feature of output to present the various details in discrete categories. These reports can be viewed on screen as well as can be kept as a hardcopy in the printed layouts. Our system produces following reports:

1. List of books
2. List of members
3. List of books issued
4. List of books returned.

Database Design

The following are various entities along with attributes for the project:

Book_Details -(Bookld, Name, Author, Category, Subject, Price, Date, Copies, Edition).
Book- Copy -(BookId, BookIDC).
Book_lR -(l1D, Bookld, Memld, Mode, State).
Book- TRDelail- (l1D, Issue_Date, Issue_Time, Return_Date, Return_Time). *Book_Delete* -(BookId, Date).
Book_Resumed -(Book_Id, Date).
Book_State -(BookId, Status).
Member_Details - (MemID, First, Middle, Last, Sex, Address, City, State, Country, Pin, Age, Phone, Issue Limit, Date_of_Joining, Books_Issued).
Member_Date -(MemID, State).
Member_Delete -(MemlD, Date).
Member_Resumed -(MemlD, Date).

After these steps, coding in any programming languages can be done and then the system will be tested against the requirements of the user. The tested system will be implemented either by direct conversion or by parallel conversion.

SELF ASSESSMENT EXERCISE

What is the main idea behind Joint Application Design?

4.0 CONCLUSION

In this unit, you learned about the systems development life cycle and other frameworks which have been developed to address the life cycles problems. These alternative frameworks include:

Prototyping (Rapid Application Development Approach)
Joint Application Design
Participating Design

5.0 SUMMARY

What you have leaned in this unit borders on the products of the SDLC phases.

6.0 TUTOR-MARKED ASSIGNMENT

What do you understand by Prototyping?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

Kendall & Kendall; *System Analysis and Design*; (Fifth Edition).PHI.

Online Resources

http://www.rspa.com
http://www.leee.org

UNIT 9 DOCUMENTATION OF SYSTEMS

CONTENTS

1.0 INTRODUCTION

This unit introduces concepts and process of documentations. It further delves into the types of documentation.

2.0 OBJECTIVES

By the end of this unit, you should be able to:

understand the concept of documentation

be aware of the steps involved in the process of documentation
describe the types of documentation.

3.0 MAIN CONTENT

3.1 Concepts and Process of Documentation

Documentation may be defined as the process of communicating about the system. The person who is responsible for this communication is called documenter. It may be noted that documenter is not responsible for the accuracy of the information, and his job is just to communicate or transfer the information.

The ISO standard ISO/IEC] 2207:1995 describes documentation “as a supporting activity to record information produced by a system development life cycle process.”

Why documentation?

Documentation is needed because it is

a means for transfer of knowledge and details about description of the system
to communicate among different teams of the software project;
to help corporate audit" and other requirements of the organization;
to meet regulatory demand;
needed for IT infrastructure management and maintenance; and
needed for migration to Ii new software platform.

Document communicates the details about the system targeted at different audience. It explains the system. The ever-increasing complexity of information system requires emphasis on well-established system of documentation. Every information system should be delivered along with an accurate and understandable document to those who will use the software.

Traditionally, documentation was done after the development of the software is completed. However, as the software development process is becoming complex and involved, documentation has become an integral part of each system development process. Documentation is now carried out at every stage as a part of development process. We will also discuss how documentation affects quality of the software later in this section.

When the process of documentations undertaken as a separate process, it requires planning in its-own right. The figure below shows, how at the development process, documentation is done alongside each step. Design and development activities of software depend on a certain base document. Documentation is to be carried out before actually implementing the design. In such a case, any flaw in design identified can be changed in the document thereby saving cost and time during implementation. If documentation is being developed for an existing software, then documentation is done along side the software development process.

The Process of Documentation

The following are various steps involved in the process of documentation:

Collection of Source Material: The very first step of any documentation process is to acquire the required source material for preparation of document. The material is collected including specifications, formats, screen layouts and report layouts. A copy of the operational software is helpful for preparing the documentation for user.

Documentation Plan: The documenter is responsible for preparation of a documentation plan, which specifies the details of the work to be carried out to prepare the document. It also defines and the target audience.

Review of Plan: The plan as set out in the process above is reviewed to see that material acquired is correct and complete.

Creation of Document: The document is prepared with the help of document generator.

Testing of Document: The document created is tested for usability as required by the target audience.

Maintain Document: Once the document is created and distributed, it must be kept up to date with new version of the software product. It must be ensured that the latest document is available to the user of the software.

Figure 4:1 depicts various stages of the process of documentation.

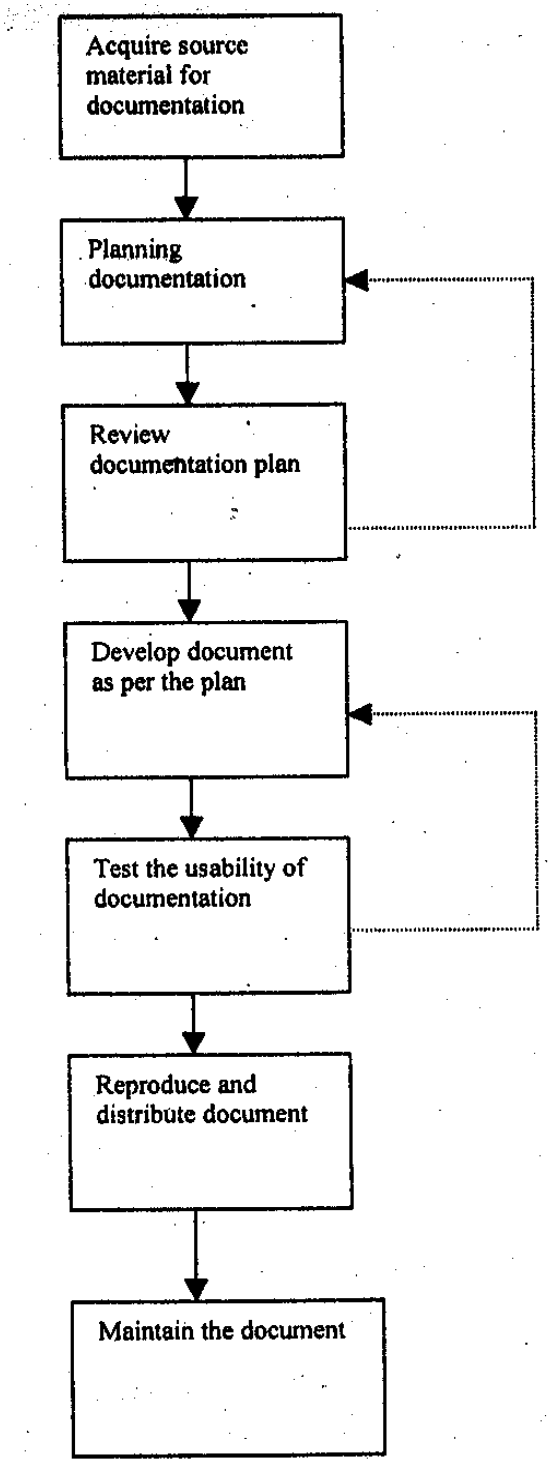


Figure 4.1: Stages of process of documentation

3.2 Types of Documentation

Any software project is associated with a large number of documents depending on the complexity of the project. Documentation that are associated with system development has a number of requirements. They are used by different types of audience in different ways as follows:

They act as a means of communication between the members of development team
Documents are used by maintenance engineer
Documents are used by the user for operation of the software
Documents are used by system administrator to administer the system.

3.2.1 System Requirements Specification

System requirement specification is a set of complete and precisely stated properties along with the constraints of the system that the software must satisfy. A well designed software requirements specification establishes boundaries and solutions of system to develop useful software. All tasks, however minute, should not be underestimated and must form part of the documentation.

Requirements of SRS: The SRS should specify only the external system behaviour and not the internal details. It also specifies any constraints imposed on implementation. A good SRS is flexible to change and acts as a reference tool for system developer, administrator and maintainer.

Characteristics of a System Requirements Specification (SRS)

1. All the requirements must be stated *unambiguously*. Every requirement stated has only one interpretation. Every characteristic of the final product must be described using a single and unique term.

2. It should be *complete.* The definition should include all functions and constraints intended by the system user. In addition to requirements of the system as specified by the user, it must conform to any standard that applies to it.

3. The requirements should be *realistic* and achievable with current technology. There is no point in specifying requirements which are unrealisable using existing hardware and software technology. It may be acceptable to anticipate some hardware

developments, but developments in software technology are much less predictable.

4. It must be *verifiable and consistent.* The requirements should be shown to be consistent and verifiable. The requirements are verified by system tester during system testing. So, all the requirements stated must be verifiable to know conformity to the requirements. No requirement should conflict with any other requirement.

5. It should be *modifiable*. The structure and style of the SRS are such that any necessary changes to the requirements can be made easily, completely and consistently.

6. It should be *traceable* to other requirements and related documents. The origin of each requirement must be clear. The SRS should facilitate the referencing of each requirement for future development or enhancement of documentation. Each requirement must refer to its source in previous documents.

7. SRS should not only address the explicit requirement but also implicit requirements that may come up during the maintenance phase of the software. It must be *usable* during operation and maintenance phase. The SRS must address the needs of the operation and maintenance phase, including the eventual replacement/of the software.

Rules for Specifying Software Requirements

The following are the rules for specifying software requirements:

Apply and use an industry standard to ensure that standard formats are used to describe the requirements. Completeness and consistency between various documents must be ensured.

Use standard models to specify functional relationships, data flow between the systems and sub-systems and data structure to express complete requirements.

Limit the structure of paragraphs to a list of individual sentences to increase the tractability and modifiability of each requirement and to increase the ability to check for completeness. It helps in modifying the document when required.

Phrase each sentence to a simple sentence. This is to increase the verifiability of each requirement stated in the document.

Structure of a Typical SRS Document:

1. Introduction

System reference and business objectives of the document.
Goals and objectives of the software, describing it in the context of the computer-based system.
The scope of the document.

2. Informative Description about the System

Information flow representation.
Information content and structure representation.
Description of sub-systems and System interface.
A detailed description of the problems that the software must solve.
Details of Information flow, content, and structure are documented.
Hardware, software, and user interfaces are described for external system.

3. Functional Description of the System

Functional description.
Restrictions/limitations.
Performance requirements.
Design cortstraints.
Diagrams to represent the overall structure of the software graphically.

4. Test and Validation Criteria

Performance limitation, if any.
Expected software response.
It is essential that time and attention be given to this section.

5. Glossary

Definitions of all technical or software-specific terms used in the document.

6. Bibliography

List and reference of all documents that relate to the software.
Supplementary information to the specification.

3.2.2 System Design Specification

The system design specification or software design specification as referred to have a primary audience, the system implementer or coder. It is also an important source of information for the system verification and testing. The system design specification gives a complete understanding of the details of each component of the system, and its associated algorithms, etc.

The system design specification documents all as to how the requirements of the system are to be implemented. It consists of the final steps of describing the system in detail before the coding starts.

The system design specification is developed in a two stage process: In the first step, design specification generally describes the overall architecture of the system at a higher level. The second step provides the technical details of low-level design, which will guide the implementer. It describes exactly what the software must perform to meet the requirements of the system.

Tools for Describing Design

Various tools are used to describe the higher level and lower level aspect of system design. The following are some of the tools that can be used in the System Design Specification to describe various aspects of the design.

Data Dictionary

Definition of all of the data and control elements used in the software product or sub-system. Complete definition of each data item and its synonyms are included in the data dictionary. A data dictionary may consist of description of data elements and definitions of tables,

Description of Data Element

Name and aliases of data item (its formal name).
Uses (which processes or modules use the data item; how and when the data item is used).
Format (standard format for representing the data item).
Additional information such as default values, initial value(s), limitations and constraints that are associated with the data elements.

Table Definitions

Table name and Aliases.

Table owner or database name.
Key order for all the tables, possible keys including primary key and foreign key.
Information about indexes that exist on the table.

Database Schema: Database schema is a graphical presentation of the whole database. Data retrieval is made possible by connecting various tables through keys. Schema can be viewed as a logical unit from programmer's point of view.

E-R Model: Entity-relationship model is database analysis and design tool. It lists real-life application entities and defines the relationship between real life entities that are to be mapped to database. E-R model forms basis for database design.

Security Model: The database security model associates users, groups, of users or applications with database access rights.

Trade-Off Matrix

A matrix that is used to describe decision criteria and relative importance of each decision criterion: This allows comparison of each alternative in a quantifiable term.

Decision Table

A decision table shows the way the system handles input conditions and subsequent actions on the event. A decision table is composed of rows and columns, separated into four separate quadrants.

Input Conditions	Condition Alternatives
Actions	Subsequent action Entries

Timing Diagram

Describes the timing relationships among various functions and behaviours of each component. They arc used to explore the behaviours of one or more objects 1hroughout a given period of time. This diagram is specifically useful to describe logic

State Machine Diagram

State machine diagrams are good at exploring the detailed transitions between states as the result of events. A state machine diagram is shown in figure 4.2.

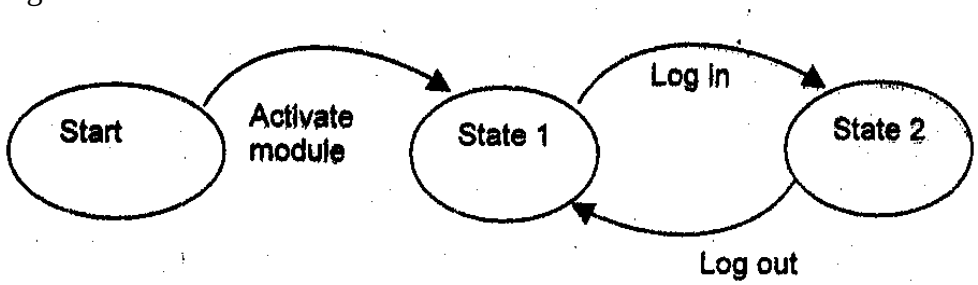


Figure 4.2: A state machine diagram

Object Interaction Diagram

It illustrates the interaction between various objects of an object-oriented system. Please refer to figure 4.3. This diagram consists of directed lines between clients and servers. Each box contains the name of the object. This diagram also shows conditions for messages to be sent to other objects. The vertical distance is used to show the life of an object.

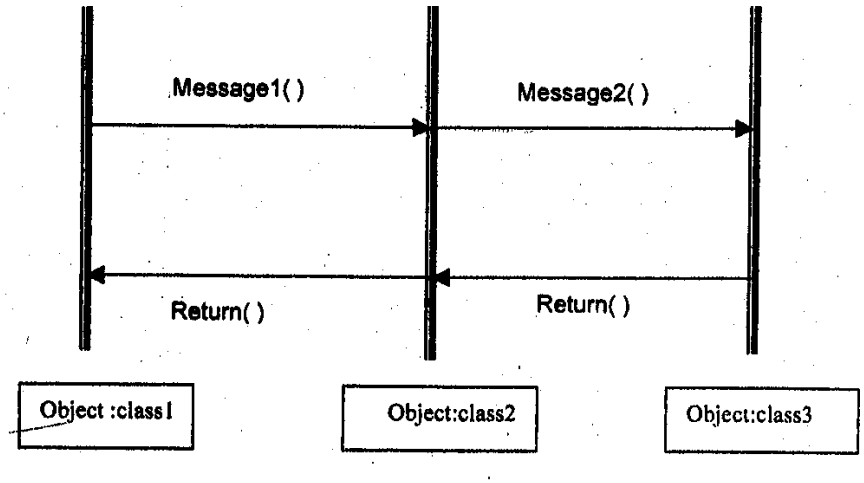


Figure 4.3: An object interaction diagram

A flow chart shows the flow of processing control as the program executes. Please refer to figure 4.4.

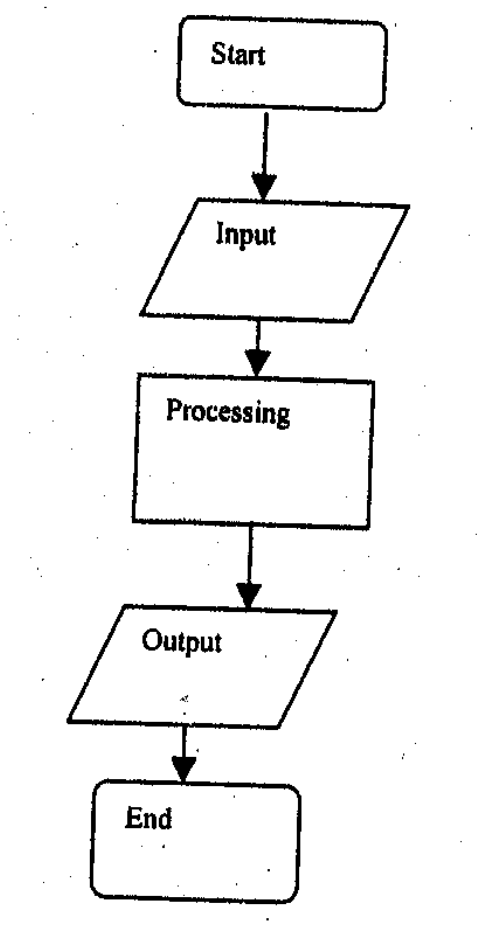


Figure 4.4: Flow Chart

Inheritance Diagram

It is a design diagram work product that primarily documents the inheritance relationships between classes and interfaces in object-oriented modeling. The standard notation consists of one box for each class. The boxes are arranged in a hierarchical tree according to their inheritance characteristics. Each class box includes the class name, its attributes, and its operations. Figure 4.5 shows a typical inheritance diagram.

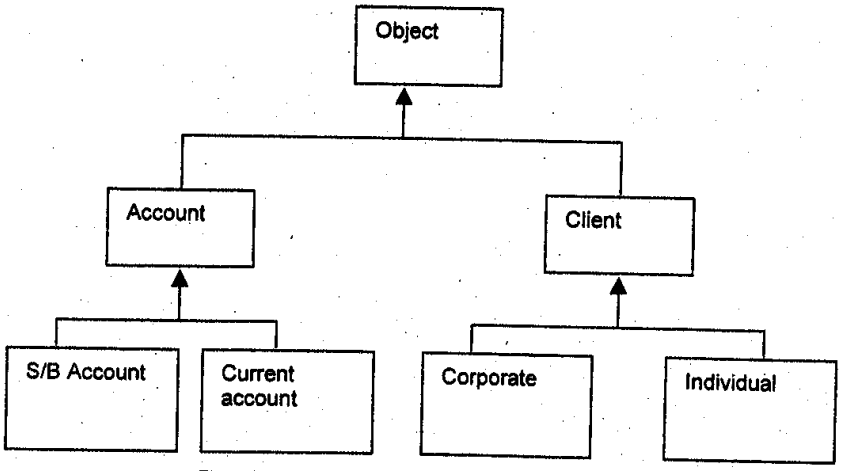


Figure 4.5: A typical inheritance diagram

Aggregation Diagram

The E-R model cannot express relationships among relationships. An aggregation diagram shows relationships among objects. When a class is formed as a collection of other classes, it is called an aggregation relationship between these classes. Each module will be represented by its name. The relationship will be indicated by a directed line from container to container. The directed line is labelled with the cardinality of the relationship. It describes "has a" relationship. Figure 4.6 shows an aggregation between two classes (circle *has a* shape).

Figure 4.6: Aggregation

Structure Chart

A structure chart is a tree of sub-routines in a program (Refer to figure 4.7). It indicates the interconnections among the sub-routines. The sub-routines should be labelled with the same name used in the pseudo code.

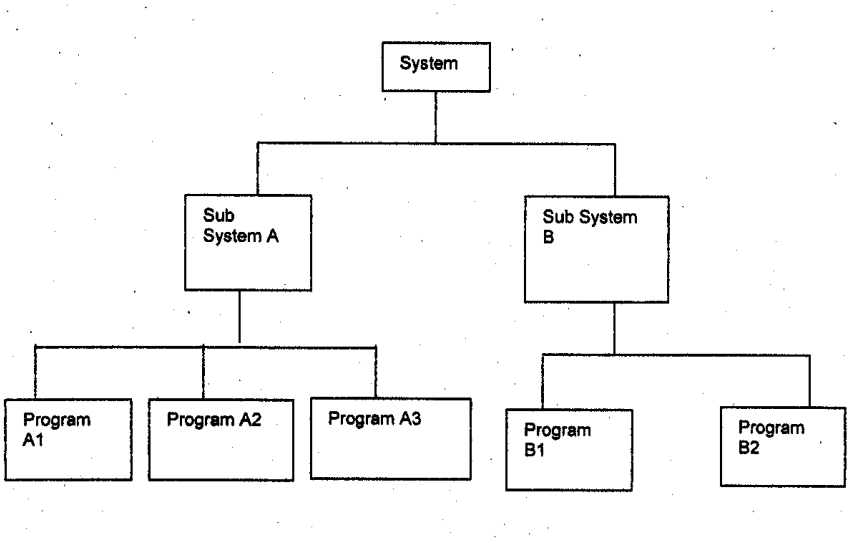


Figure 4.7: A structured chart

Pseudocode

Pseudocode is a kind of structured English for describing algorithms in an easily readable and modular form. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax in which the code is going to be written. The pseudocode needs to be complete.

It describes the entire logic of the algorithm so that implementation becomes a routine mechanical task of translating line by line into source code of the target language. Thus, it must include flow of control.

This helps in describing how the system will solve the given problem. "Pseudocode" does not refer to a precise form of expression. It refers to the simple use of Standard English. Pseudocode must use a restricted subset of English in such a way that it resembles a good high level programming language. Figure 4.8 shows an example of pseudo code.

```
IF Hours Worked > Max Work Hour THEN
        Display overtime message

ELSE
        Display regular message

ENDIF
```

Figure 4.8: Example of a Pseudocode

Contents of a typical System Design Specification document content

1. Introduction

Purpose and scope of this document:
Full description of the main objectives and scope of the SDS document is specified.
Definitions, acronyms, abbreviations and references
Definitions and abbreviations used are narrated in alphabetic order. This section will include technical books and documents related to design issues. It must refer to the SRS as one of the reference book.

2. System Architecture Description

Overview of modules, components of the system and sub-systems
Structure and relationships
Interrelationships and dependencies among various components are described.
Here, the use of structure charts can be useful.

3. Detailed Description of Components

Name of the component .Purpose and function Sub-routine and constituents of the component
Dependencies, processing logic including pseudocode
Data elements used in the component.

4. Appendices

3.2.3 Test Design Document

During system development, this document provides the information needed for adequate testing. It also lists approaches, procedures and standards to ensure that a quality product that meets the requirement of the user is produced. This document is generally supplemented by documents like schedules, assignments and results. A record of the final result of the testing should be kept externally.

This document provides valuable input for the maintenance phase.

The following IEEE standards describe the standard practices on software test and documentation:

1. 829-1998 IEEE Standard for Software Test Documentation
2. 1008-1987 (RI993) IEEE Standard for Software Unit Testing

3. 1012-1998 IEEE Standard for Software Verification and Validation

The following is the typical content of Test Design Document:

1. Introduction

Purpose

The purpose of this *document* and its intended audience are clearly stated.

Scope

Give an overview of testing process and major phases of the testing process. Specify what is not covered in the scope of the testing such as, supporting or not third party software.

Glossary

It gives definition of the technical terms used in this document.

References

Any references to other external documents stated in this document including references to related project documents. They usually refer the System Requirement Specification and the System Design Specification documents.

Overview of Document

Describe the contents and organization of the document.

2. Test Plan

A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, and the person who will do each task, and any risks that require contingency planning.

Schedules and Resources

An overview of the testing schedule in phases along with resources required for testing is specified.

Recording of Tests

Specify the format to be used to record test results. It should very specifically name the item to be tested, the person who did the testing, reference of the test process/data and the results expected by the test, the date tested. If a test fails, the person with the responsibility to correct and retest is also documented. The filled out format would be kept with the specific testing schedule. A database could be used to keep track of testing.

Reporting Test Results

The summary of what has been tested successfully and the errors that still exist which are to be rectified is specified.

3. Verification Testing

Unit Testing

For each unit/component, there must be a test which will enable tester to know about the accurate functioning of that unit.

1ntegration Testing

Integration test is done on modules or sub-systems.

4. Validation Testing

System Testing

This is the top level of integration testing. At this level, requirements are validated as described in the SRS.

Acceptance and Beta Testing

List test plans for acceptance testing or beta testing. During this test, real data is used for testing by the development team (acceptance testing/alpha testing) or the customer (beta testing). It describes how the results of such testing will be reported back and handled by the developers.

3.2.4 User Manual

This document is complete at the end of the software development process.

Different Types of User Documentation

Users of the system are not of the same category and their requirements vary widely. In order to cater to the need of different class of user, different types of user documentation are required. The following are various categories of manuals:

Introductory manual: How to get started with the system?
Functional description: Describes functionality of the system.
Reference manual: Details about the system facility.
System administrator guide: How *to* operate and maintain the system?
Installation document: How to install the system?

The following is the typical content of User Manual

1. Introduction

Purpose

The purpose of this *document* and its intended audience is stated. If there is more than one intended audience, provide information in this section and direct the reader to the correct section(s) for his/her interest.

Scope of Project

Overview the product. Explain who could use the product. Overview the services that it provides. Describe limitation of the system. Describe any restrictions on using or copying the software and any warranties or contractual obligations or disclaimers.

Glossary

Define the technical terms used in this document. Do not assume that the reader is expert.

References

References to other documents cited anywhere in this document including references to related project documents. This is usually the only bibliography in the document

Overview of Document

The contents and organization of the rest of this document are described.

2. Instructional Manual

This section should be divided in the manner that will make it user friendly.

System Usage

Provide examples of normal usage. Images of the screen dumps are very useful to provide a look and feel of the product. Provide any necessary background Information. On-line help system is very common for systems today. Information on how to use the on-line help system, how to access it may be provided here.

3. User Reference Manual

List of Services

Provides an *alphabetical* listing of services provided by the system with references to page numbers in this document where the concerned service is described.

Error Messages and Recovery

Provides an *alphabetical* list or all error messages generated by the system and how the user can recover from each of these errors.

4. Installation Information

Installation information is provided including the operating environment.

Maintenance Manual

The supplier/developer of the software is sometimes different from the software maintenance agency. In such cases, the criticality of maintenance manual assumes a bigger role. Maintenance manuals provide precise information to keep your product operating at peak performance. Similar to installation manuals, these documents may range from a single sheet to several hundred of pages.

SELF ASSESSMENT EXERCISE

Explain the tools used in the system design specification to describe various aspects of the design.

4.0 CONCLUSION

In this unit you have learned about he concepts, and process of documentation. You have also learned about the types of documentations.

5.0 SUMMARY

What you have studied in this unit borders on concepts abd process of documentation.

6.0 TUTOR-MARKED ASSIGNMENT

What are the rules for specifying software requirements?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

ISO/IEC 12207: Software life cycle process

IEEE 1063: Software user Documentation

ISO/IEC 18019: Guide lines for the design and preparation of user documentation for application software.

Online Resources

http://www.sce.carleton.ca/squall
http://en.tldp.org/HOW/Software-Release-Practice-HOWTO/documentation.html
http;//www.sei.cmu.edu/cmm/

UNIT 10 DOCUMENTATION STANDARDS

CONTENTS

1.0 INTRODUCTION

In this unit you will learn about the different standards for documentation as well as the quality of software. You will equally learn about the good practices for documentation.

2.0 OBJECTIVES

After going through this unit, you should be able to:

explain the different standards for documentations
describe the quality of software
discuss the good practices for documentation.

3.0 MAIN CONTENT

3.1 Different Standards for Documentation

This software documentation standard is used in the organization for uniform practices for documentation preparation, interpretation, change, and revision, to ensure the inclusion of essential requirements of different standards. Sometimes, documentation as per various standards is stated in the contractual agreement between the software vendor and the customer.

This standard will also aid in the use and analysis of the system/sub-system and its software documentation during the system/software life cycle of a software project. Documentation comes in many forms, e.g., specifications, reports, files, descriptions, plans, source code listings, change requests, etc. and can be in electronic or paper form.

The Documentation Standard defines various aspects of documentation such as style, format, and the document revision/change process of these documents.

'The International Standards, ISO/IEC 12207 -Software life cycle process, describes documentation as one of the supporting parallel process of software development process. It may be noted that this standard is not documentation standard but describes the process of documentation during the software development process. The following are other documentation standards:

ISO/EC 18019: Guidelines for the design and preparation of user documentation for application software.

This standard describes how to establish what information users need, how to determine the way in which that information should be presented to the users, and then how to prepare the information and make it available. It covers both on-line and printed documentation. It describes standard format and style to be adopted for documentation. It gives principles and recommended practices for documentation.

ISO/EC 15910: Software user documentation process

This standard specifies the minimum process for creating user documentation for software that has a user interface, including printed documentation (e.g., user manuals), on-line documentation, help text and on-line documentation systems.

IEEE 1063: Software user Documentation

It provides minimum requirement for structure, information content and format for user documentation. It does not describe the process to be adopted for documentation. It is applicable for both printed and on-line documentation.

Components of software user documentation as described In IEEE 1063: Software user Documentation:

Components of Software User Documentation

1. Identification data (e.g., Title Page)
2. Table of contents
3. List of illustrations
4. Introduction
5. Information for use of the documentation such as description of software etc.

6. Concept of operations
7. Procedures
8. Information on software commands
9. Error messages and problem resolution
10. Glossary (to make the reader acquainted with unfamiliar terms)
11. Related information sources
12. Navigational features
13. Index
14. Search capability (for electronic document).

Documentation involves recording of information generated during the process of software development life cycle. Documentation process involves planning, designing, developing, distributing and maintaining documents.

During planning phase, documents to be produced during the process .of software development are identified. For each document, following items are addressed:

Name of the document
Purpose
Target audience
Process to develop, review, produce, design and maintain.

The following form part of activities related to documentation of developing phase:

All documents to be designed in accordance with applicable documentation standards for proper formats, content description, page number, figure/table. Source and accuracy of input data for document should be confirmed.
Use of tools for automated document generation.
The document prepared should be of proper format. Technical content and style should be in accordance to documentation standards.

Production of the document should be carried out as per the drawn plan. Production may be in either printed form or electronic form. Master copy of the document is to be retained for future reference.

Maintenance

As the software changes, the relevant documents are required to be modified. Documents must reflect all such changes accordingly.

3.2 Documentation and Quality of Software

Inaccurate, incomplete, out of date, or missing documentation is a major contributor to poor software quality. That is why documentation and document control has been given due importance in ISO 9000 standards, SEI CMM software Maturity model. In SEI CMM Process Model and assessment procedure, the goal is to improve the documentation process that has been designed. A maturity level and documentation process profile is generated from the responses to an assessment instrument.

One basic goal of software engineering is to produce the best possible working software along with the best possible supporting documentation. Empirical data show that software documentation products and processes are key components of software quality. Studies show that poor quality, out of date, or missing documentation are a major cause of errors in software development and maintenance. Although everyone agrees that documentation is important, not everyone fully realizes that documentation is a critical contributor to software quality.

Documentation developed during higher maturity levels produces higher quality software.

3.3 Good Practices for Documentation

Documentation is the Design Document: The time to document is before actually implementing any design. A lot of effort can be saved in such cases.

Good documentation projects the quality of software Many people take poor, scanty, or illiterate documentation for a program as a sign that the programmer is sloppy or careless of potential users' needs. Good documentation, on the other hand, conveys a message of intelligence and professionalism. If your program has to compete with other programs, better make sure that your documentation is at least as good as your competitors.

SELF ASSESSMENT EXERCISE

List six components of software user documentation

4.0 CONCLUSION

You have learned about the different standards for documentation. You have also learned about the documentation and quality of software. Finally, you have learned about good practices for documentation

5.0 SUMMARY

What you have learned in this unit borders on the different standards for documentation and quality of software.

6.0 TUTOR-MARKED ASSIGNMENT

What do you understand by the documentation standard?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L., Whitten, Lonnie D. Bentley, Kevin C. Dittman (2001). *System Analysis and Design Methods*; (Fifth Edition).Tata: McGraw-Hill.

ISO/IEC 12207: Software life cycle process

IEEE 1063: Software user Documentation

ISO/IEC 18019: Guide lines for the design and preparation of user documentation for application software.

Online Resources

http://www.sce.carleton.ca/squall
http://en.tldp.org/HOW/Software-Release-Practice-HOWTO/documentation.html
http;//www.sei.cmu.edu/cmm/

MODULE 2 PLANNING AND DESIGNING INFORMATION SYSTEMS

UNIT 1 PROCESS OF SYSTEMS PLANNING

CONTENTS

1.0 INTRODUCTION

In this unit you will learn about the various techniques of fact finding and getting appropriate information about systems which is currently in place.

2.0 OBJECTIVES

After going through this unit, you should be able to:

mention the types of interviews
enumerate the steps to be followed for a successful interview
explain group discussions
describe how site visits are conducted

describe how presentations are done
explain the two types of questionnaires.

3.0 MAIN CONTENT

3.1 Fact Finding Techniques

To learn the functions of the existing system, systems analyst needs to collect data related to the existing system. Usually, the data related to organization, staff, documents used, formats used in the input and output processes is collected. This information is obtained through interviews, group discussions, site visits, presentations, and questionnaires.

Need for Fact Finding

Normally, each and every business house or any organization has its own rules and procedures to run and manage it. When a system needs to be developed, the systems analyst needs to know the requirements of the system. Depending on these requirements, the system has to be developed.

3.2 Interviews

Personal interview is a recognized and most important fact finding technique, where the systems analyst gathers information from individual through face to face interaction. Interviews are used to find the facts, verify facts, clarify facts, get the customer involved, identify the system requirements and know all options. The interview is usually conducted by the systems analyst. To conduct interview, the interviewer must have personality which helps him/her to be social with strangers or different types of people. Always and for all situations, interviews are not appropriate fact finding methods. It has both advantages and disadvantages.

Advantages

Interviews permit the systems analyst to get individual's views and get the specific problem work wise and operation wise.

Interviews allow the systems analyst to obtain a better clarity of the problem due to feedback from the interviewees.

In the process of interviews, the interviewer has time and scope to motivate the interviewee to respond freely and openly.

Interviews allow the systems analyst to understand the user requirements and to know the problems faced by the user with the current system.

It is an effective technique to gather information about complex existing systems.

Disadvantages

Interviews are very time consuming.

Success of interviews, in most of the cases, depends on the systems analyst's interpersonal relationship skills.

Some times, interviews may be impractical due to the location of interviewees.

Types of Interviews

There are two types of interviews:

Structured; and
Unstructured.

In structured interviews, there is a specific set of questions to be asked to an interviewee. In the case of unstructured interviews, there are few specific questions pertaining to an interviewee. But, you have questions which are common to all interviewees. Unstructured interviews are conducted with only a general goal or subject in mind.

Conducting Interview is an art. The success in interview depends on selecting the individual, preparing for the interview, creating situation in which the answers offered are reliable and creating a situation in which opinion can be given without any fear of being criticized by others.

Arranging Interview

The system analyst should prepare properly for the interview. S/he should select place of interview, time of interview in such a way so that there will be minimal interruption. Always, it is important to take appointment with the interviewee. Time to be spent during interview varies from project to project. The higher the management level of the interviewee, the less the time to be scheduled for the interview.

Guidelines for Conducting Interviews

For a successful interview, the steps to be followed are given below:

Introduction

During introduction; the analyst should introduce himself by focusing on purpose of the interview and the confidential nature of interview. Also, this is the phase wherein first impressions are formed and pave way for the success of the remaining part of the interview.

Asking Questions

Questions should be asked exactly as these are worded in case of structured interview. Rewording may modify or bias the response. Always, questions have to asked in the same sequence as prepared.

Recording the Interview

Record of the interview must be kept mentioning the source of the data and its time of collection. Sometimes, the analyst cannot remember the source of the data which may attribute to the invalid sources.

Doing a Final Check

After the interview has been completed, the deliberations made during the interview should be put in the form of a report. The report of the interview has to be sent to the interviewee for his/her signature, If any discrepancies are found or any modifications are to be done, these CaJ1 be done at this point of time.

3.3 Group Discussions

In this method, a group of staff members are invited who are expected to be well versed in their own wings of the organization. The analysts will have a discussion with the members for their views and responses to various queries posed by them.

In this process, individuals from different sections gather together and will discuss tl1e problem at hand. Ultimately, tl1ey come to an optimum solution. In this process, the problems of all sections are taken care of most of the cases; solutions are found which are acceptable to everyone. The main disadvantage of this process is that it is very difficult to get all the concerned people together at a time. But, the major advantage is that a mutually acceptable solution can be found.

3.4 Site Visits

The engineers of the development organization visit the sites. Usually, the systems analysts visit sites to get first hand information of the working of the system. In this technique, systems analyst watches the activities of different staff members to learn about the system. When there is confusion about the validity of data collected from other sources, the systems analyst uses the method of site visits. The main objective of site visit is to examine the existing system closely and record the activities of the system.

Advantages

1. The process of recording facts site visits is highly reliable.
2. Sometimes, site visits take place to clear doubts and check the validity of the data.
3. Site visit is inexpensive when compared to other fact finding techniques.
4. In this technique, systems analyst will be able to see the processes in the organization at first hand.
5. The systems analyst can easily understand the complex processes in the organization.

Disadvantages

1. People usually feel uncomfortable when being watched; they may unwillingly perform their work differently when being observed.
2. Due to interruptions in the task being observed, the information that is collected may be inaccurate.
3. Site visits are done during a specific period and during that period, complexities existing in the system may not be experienced.
4. There may be scheduling problems for the systems analysts when the activities take place during odd hours.
5. Sometimes, people may be more careful to adopt the exact procedure which they do not typically follow. .

Guidelines for Site Visit

Site visits are to be conducted where the work load is normal. After studying the work and normal work load, systems analyst can observe the work at peak hours to see the effect caused by increased volumes. The systems analyst should collect the input/output form, documents at the time of his/her visit. The following guidelines need to be followed at the time of observation and site visit:

1. Keep a low profile at the time of site visit.
2. Take necessary permissions from appropriate officials to conduct site visit.
3. Inform the individuals who will be observed at the time of site visit.
4. Take notes of the study of site visit immediately.
5. Do not make any assumptions.

3.5 Presentations

It is another way of finding the facts and collecting data. Presentation is the way by which the systems analyst gathers first hand knowledge of the project. The customer makes a presentation of the existing system or about the organization. Participants in the meeting are representatives from the IT company and key personnel of the client organization. When a company needs to develop a software project, it may present its requirements for IOE (interest of expression) from the interested IT Company. In that case, the client presents his/her requirements. Based on the requirements, tile IT companies make prototype and show the demo of the prototype. It is very difficult to obtain information in detail from a presentation. But, information available through presentation is sufficient to develop a prototype. Presentation is made by the concerned department in consultation from other departments and senior officials.

3.6 Questionnaires

Questionnaires are special purpose documents that allow the analyst to collect information and opinion from respondents. By using questionnaires, it is possible to collect responses or opinion from a large number of people. This is the only way to get response from a large audience.

Advantages

1. It is an inexpensive means of collecting the data from a large group of individuals
2. It requires less skill and experience to administer questionnaires.
3. Proper formulation and interaction with respondents leads to unbiased response from the customers. .
4. Customers can complete it at their convenience.
5. Responses can be tabulated and analyzed quickly.

Disadvantages

1. Sometimes, the number of respondents is low.
2. There is no guarantee that the respondents will answer all the questions.
3. Sometimes, the individual may misunderstand the question. In that situation, the analyst may not get correct answer.

Types of Questionnaires

There are two types of questionnaires:

Free formed questionnaires are questionnaires where questions are mentioned along with blank spaces for response.

Fixed formed questionnaires are questionnaires which consist of multiple choices and the respondent can select only from the choices provided.

SELF ASSESSMENT EXERCISE

1. What are the guidelines for conducting interviews?
2. Discuss the two types of questionnaires

4.0 CONCLUSION

In this unit, you have learned about the various techniques of fact finding and how to get appropriate information about a system currently in place.

5.0 SUMMARY

What you have learned from this unit is based on fact finding techniques.

6.0 TUTOR-MARKED ASSIGNMENT

Explain the two types of interviews.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorey F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award;(1994). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com
http;//www.cbpa.edu/flin/info609/sysplan

UNIT 2 FEASIBILITY STUDY

CONTENTS

1.0 INTRODUCTION

The need to evaluate the feasibility of a project at the earliest possible time is discussed in this unit.

2.0 OBJECTIVES

After going through this unit, you should be able to:

understand the technical feasibility of a project
be aware of the operational feasibility of a project
describe the economic feasibility of a project
explain the legal feasibility of developing a system.

3.0 MAIN CONTENT

3.1 Issues Involved in Feasibility Study

Feasibility study consists of activities which determine the existence of scope of developing an information system to the organization. This study should be done throughout the life cycle. In a project, at one point of time, it may seem that the project is feasible. But, after proceeding one or two phases, it may become infeasible. So, it is necessary to evaluate the feasibility of a project at the earliest possible time. Months or years of efforts, huge finances could be saved if an infeasible system is recognized at earlier stage.

Feasibility study starts from the preliminary investigation phase. At this stage, the analyst estimates the urgency of the project and estimates the development cost.

The next check point is problem analysis. At this stage, the analyst studies current system. S/he does it to understand the problem in the better way. It helps him/her to make better estimates of development cost, and also to find out the benefits to be obtained from the new system. In feasibility analysis, we have to study the following:

Technical feasibility,
Operational feasibility,
Economic feasibility, and
Legal Feasibility.

3.2 Technical Feasibility

Technical feasibility is concerned with the availability of hardware and software required for the development of the system, to see compatibility and maturity of the technology proposed to be used and to see the availability of the required technical manpower to develop the system. These three issues are addressed during this study.

Is the proposed technology proven and practical? At this stage, the analyst has to see or identify the proposed technology, its maturity, its ability or scope of solving the problem. If the technology is mature, if it has large customer base, it will be preferable to use as large customer base already exists and problems that stem from its usage may be less when compared to other technologies which don't have a significant customer base. Some companies want to use the state of art new technology irrespective of the size of customer base.

The next question is: does the firm possess the necessary technology it needs. Here, we have to ensure that the required technology is practical, and available. Now, does it have the required hardware, and software? For example, we need ERP software, and hardware which can support ERP. Now, if our answer is no for either of the questions, then the possibility of acquiring the technology should be explored.

The last issue related to technical feasibility is the availability of technical expertise. In this case, Software and Hardware are available. But, it may be difficult to find skilled manpower. The company might be equipped with ERP software, but the existing manpower might not have the expertise in it. So, tile manpower should be trained in the ERP software. This may lead to slippage in the delivery schedules.

3.3 Operational Feasibility

Operational feasibility is all about problems that may arise during operations. There are two aspects related with this issue:

What is the probability that tile solution developed may not be put to use or may not work?

What is the inclination of the management and end users towards the solution? Though. there is very least possibility of management being averse to the solution, there is a significant probability that the end users may not be interested in using the solution due to lack of training, insight, etc.

Also, there are other issues related with operational feasibility.

Information

The system needs to provide adequate, timely, accurate and useful information. It should be able to supply all the useful and required information to all levels and categories of users.

Response time

It needs to study the response time of the system in term of throughput. It should be fast enough to give the required output to the users. .

Accuracy

A software system must operate accurately. It means that it should provide value to its users. Accuracy is the degree to which the software performs its required functions and gives desired output correctly.

Security

There should be adequate security to information and data. It should be able to protect itself from fraud.

Services

The system needs to be able to provide desirable and reliable services to its users.

Efficiency

The system needs to be able to use maximum of the available resources in an efficient manner so that there are no delays in execution of jobs.

3.4 Economy Feasibility

It is tile measure of cost effectiveness of tile project. The economic feasibility is nothing but judging whether the possible benefit of solving the problems is worthwhile or not. At the feasibility study level, it is impossible to estimate the cost because customer's requirements and alternative solutions have not been identified at this stage. However, when the specific requirements and solutions have been identified, the analyst weighs the cost and benefits of all solutions, this is called "cost benefit analysis". This is discussed below. A project which is expensive when compared to the savings that can be made from its usage, then this project may be treated as economically infeasible.

3.5 Legal Feasibility

Legal feasibility studies issues arising out of the need to tile development of the system. The possible consideration might include copyright law, labour law, antitrust legislation, foreign trade, regulation, etc. Contractual obligation may include the number of users who will be able to use the software. There may be multiple users' licenses, single user licenses, etc. Legal feasibility plays a major role in formulating contracts between vendors and users. If the ownership of the code is not given to the user, it will be difficult to install it without proper permission to other systems.

Another important legal aspect is that whenever an IT company and the user company do not belong to the same country then the tax laws, foreign currency transfer regulations, etc., have to be taken care of.

SELF ASSESSMENT EXERCISE

1. Explain the operational feasibility.
2. What is the last issue related to technical feasibility?

4.0 CONCLUSION

Issues involved infeasibility studies were discussed in this unit. Hence, technical operational, economic and legal feasibility were all considered in this unit.

5.0 SUMMARY

What you have learned in this unit focuses on the issues involved in feasibility studies.

6.0 TUTOR-MARKED ASSIGNMENT

Enumerate four issues studied in feasibility analysis

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com
http;//www.cbpa.edu/flin/info609/sysplan

UNIT 3 ANALYSIS OF THE SYSTEM

CONTENTS

1.0 INTRODUCTION

In this unit, we will consider the cost benefit analysis. We will equally consider preparing schedule as well as gathering requirements of a system.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

be aware of the cost prepare schedule
understand how to prepare schedules
know how to gather requirements of a system.

3.0 MAIN CONTENT

3.1 Cost Benefit Analysis

In economic feasibility, cost benefit analysis will be done. There are two types of costs associated with a project: The costs involved with development of the system and costs associated with operation and maintenance of the system. System development cost can be estimated at the time of planning of the system and it should be refined in different phases of the project. Maintenance and operation costs are to be estimated before hand. At the same time, these estimations are bound to change as the requirements change during the development process. After the implementation, these costs may increase or decrease depending on the nature of updations done to the system. System development cost is one time cost, but maintenance and operating costs are recurring costs. Different costs are:

Cost of Human Resources

It includes the salaries of system analysts, software engineers, programmers, data entry operators, operational, and clerical staff. In other words, the amount that is going to be spent on all the people involved.

Cost of Infrastructure

The cost of infrastructure including those of computers, cables, software, etc., comes under this head.

Cost of Training

Both the developing staff and operating staff need to be trained for new technologies and new system. So, tile training cost has to be considered for calculating the cost of the system.

There are two components in economic feasibility: *costs and benefits.* The cost consists of tangible hardware, software costs, cost of human resources and some intangible costs. Tangible costs are saved by the usage of the system. Intangible costs are saved by the quality of the system. Also, application of system should lead to efficiency. When the quality of the system is high, the effectiveness of the services provided by the organizations increase. If a choice has to be made between efficiency and effectiveness then it is better to do the right thing inefficiently than to do wrong thing efficiently. The tangible benefits are those which can be quantified easily. They can be measured in terms of savings or profits. On the other hand, in the case of intangible benefits, it is difficult to quantify. Examples of intangible benefits are improving company goodwill, improving employee moral, better decision making, etc.

We may consider the case of an insurance company's branch office. There are 15 employees in the office which include one manager, two business development officers, one accounts officer, one administrative officer, seven clerical staffs, two security guards, one peon. If the branch is converted to a fully computerized branch, the total hardware and software cost will be Rs.10 lakhs. Training of the existing manpower will be Rs.50,000. Total investment is 10.5 lakhs. Total maintenance cost of software and hardware is Rs.I.5 lakhs per year. Interest of the investment is Rs.I,26,000 per year. So, the total expenditure is increased by Rs.2, 76,000 per year. But the branch can reduce the clerical staff. Now it needs two clerical staffs and two data entry operators. Total cost saving by reducing 3 staffs will be Rs.4.5 lakhs per year. Here, it is clear that the tangible benefit is more than the expenditure. Besides, the tangible benefit also improves the customer's satisfaction. So, it is clear that tile project is feasible.

3.2 Preparing Schedule

A system development process scheduling is an activity that distributes estimated effort according to the planned project duration by allocating the effort to specific software engineering tasks. But, at the early stage of the project, macroscopic schedule is developed: This schedule identifies all major activities of the project. As the project progresses, each entity of macroscopic schedule is refined into a detailed schedule. For a systems development, scheduling is meant for setting an end date to the project(s).

Now, the feasibility of following the schedule is directly related to the time table made. Systems analysts have to take care of schedule feasibility of the system. The purpose of schedule feasibility is to understand the time frames and dates of completion of different phases of the project. It means that the project can be completed and be operational so that it will meet the needs of the user requirements.

In most cases, missing the deadline may invite penalties. A systems analyst has to remember the schedule feasibility at the time before entering into any agreement with client regarding the delivery schedules. At the project planning stage, feasibility of conforming to the schedule will be studied by the analyst. To take a decision, factors such as expected team size, availability of resources, sub-contracting or outsourcing of activities have to be considered. Scheduling feasibility will be reassessed during the commencement of each phase.

3.3 Gathering Requirements of System

Finalizing the requirements of the system be built forms the backbone for the ultimate success of the project. It not only includes ascertaining the functions, but also the constraints of the system. The later part is very important as the customer needs to be very clear about the services that are going to be offered by system. This will avoid any conflicts during the delivery or intermediate meetings with the client as the client assumes that the system provides those functions which are actually constraints of the system.

When the requirements of the system are inaccurate, it may lead to the following problems:

1. Delivery schedules may be slipped.
2. Developed system may be rejected by the client leading to the loss of reputation and amount spent on the project.
3. System developed may be unreliable.
4. Overall cost of the project may exceed the estimates.

There are different ways of finding the system requirements. Two of them are joint application development and prototyping.

3.3.1 Joint Application Development

It is a process in which group meetings are held to analyze the problem and define the requirements of the desired system. In the process of Joint Application Development (JAD), each participant is expected to attend and actively participate. The group includes: sponsor, the facilitator, the user manager and IT staff. When JAD technique is used to find the requirements, it is known as Joint Requirements Planning.

Participants of Joint Application Development

The following are the various participants of Joint Application Development:

Sponsor is a person in top management. The sponsor plays a vital role in the process of JAD. S/he works closely with JAD leader to plan the session by identifying individuals from the user community.

Facilitator is a single individual who plays an important role as leader. S/he leads all the session that held for system development. S/he must have good communication skill, negotiation skill, ability to remove group conflicts, possess good knowledge of business, has strong organizing power, quick and impartial decision making capability. The facilitator plans session for JAD, conducts the session and follows the decision of the session.

Representatives of the Clients will also attend the JAD session. They are chosen by the project sponsor based on their knowledge of the business system. The role of the representatives of the clients is to communicate the business rules and the requirements of the desired system.

Scribe records proceedings of the meeting. The proceedings are published and demonstrated to the attendees immediately after the meeting. Scribes need to have a good knowledge of systems analysis Systems analysts frequently play this role.

IT Staff such as programmer also participate in tile session. IT staff listen and take notes regarding issues and requirements mentioned by the clients and analysis. They can contribute their ideas related to technical limitations of tile current system.

JAD session spans from 3-7 days, but in special cases, it may continue up to two weeks. Success of JAD session depends upon proper planning.

For a successful JAD T session, all the participants should be informed about the schedule of the session before hand and they should come prepared. The analyst must work closely with the sponsor to determine the scope of the issues that will be discussed in JAD session. There are three steps that are to be followed for a successful JAD session:

Selection of a location for JAD session.
Selection of JAD participants.
Preparation of agenda items for JAD session.

JAD sessions are usually held in a location different from the work place. The meeting room should be equipped with white board, overhead projector, data projector, laptop, printer, scanner, etc. There should be name tags for all the participants.

Preparation of the agenda is the key for the success of JAD session. Agenda must be brief, should mention the objective of the session. It must mention the item to be discussed in each session and time allotted to each item. Agenda contains three parts namely, the opening, the body and conclusion.

Process of Conducting a JAD Session Successfully

For successful session, the facilitator should adhere to the following guidelines:

1. Agenda should be followed strictly.
2. Topic should be completed within allotted time.
3. Ensure that the scribe is able to take notes.
4. Avoid the use of technical jargon unless essential.
5. Try to get group consensus.
6. Ensure that the participants follow the rules.

Disadvantages of Joint Application Development (JAD)

Since it is a meeting of many people, there may not be sufficient time for everyone to speak.

Only a few people may dominate the discussion. So, the outcome of tile meeting will be the view of those who spoke most during the meeting.

The problem with such meetings is that some people are afraid to speak out for fear that they may be criticised.

3.3.2 Prototyping

Designing and building a scaled down, but functional version of a desired system is known as prototype. In other words, it is the model of the software to be built. It can be developed using appropriate software such as 3GL, 4GL with query, screen, report form, etc. The analyst builds a prototype as per the preliminary or basic requirements of the user. Whenever the prototype is displayed to the clients, they give their suggestions regarding improvement of features, etc., or they may accept it. Of course there is every possibility of rejection also. Based on the user feedback, the analyst improves tile prototype and makes a new version of the prototype. This process continues till the client is satisfied and fulfils his/her needs. In some cases, prototypes are further scaled upwards to become full fledged software to be delivered to tile customer. This model is useful for determining requirements for the software to be built in the following situations:

1. Requirements are not clear.
2. For any complex systems, prototypes are more useful.
3. In the cases where communication problems exist between customer and analysts, this model is useful.
4. Tools and data are readily available for building the working system.

There are some disadvantages of the prototype model:

1. In case of prototyping, formal documentation is avoided.
2. Usually, prototypes are stand alone systems. Building prototypes is difficult incases where data has to be shared.
3. Important issues, such as security and validation, are not given importance

4.0 CONCLUSION

The benefit consists of serving the tangible costs by using the system and the tangible costs by improving the quality of service. There are several modern information gathering techniques used by the systems analyst. Some of them are:

Joint Application Development (JAD) and
Prototyping

5.0 SUMMARY

In this unit, you would have learned about the analysis of a system. Specifically, the cost benefits analysis.

SELF ASSESSMENT EXERCISE

1. What is prototyping?
2. What are the two components in economic feasibility?

6.0 TUTOR-MARKED ASSIGNMENT

Describe two types of costs associated with a project.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 4 PRINCIPLES OF DESIGN

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn the process of design system's internals. We will consider design principles. Also to be considered are the top down and bottom up designs.

2.0 OBJECTIVES

After going through this unit you should be able to:

understand design principles
be aware of the top down design
describe the bottom up design.

3.0 MAIN CONTENT

3.1 Design Principles

There are certain principles that can be used for the development of the system. These principles are meant to effectively handle the complexity of process of design. These principles are:

Problem Partitioning: It is concerned with partitioning the large problems. *Divide and Conquer* is the policy adopted here. The system is divided into modules that are self dependent. It improves the efficiency of the system. It is necessary that all modules have interaction between them.

Abstraction: It is an indispensable part of design process and is essential for problem partitioning. Abstraction is a tool that permits the designer to consider a component at an abstract level (outer view)

without worrying about details of implementation of the component. Abstraction is necessary when the problem is divided into smaller parts so that one can proceed with one design process effectively and efficiently. Abstraction can be functional or data abstraction. In functional abstraction, we specify the module by the function it performs. In data abstraction, data is hidden behind functions operations. Data abstraction forms the basis for object-oriented design.

Design principles are necessary for efficient software design. Top down and bottom up strategies help implement these principles and achieve the objectives.

A system consists of components called modules, which have subordinate modules. A system is a hierarchy of components and the highest-level module called super ordinate module corresponds to the total system. To design such a hierarchy, there are two approaches namely top down and bottom up approaches.

The top down approach starts from the highest-level module of the hierarchy and proceeds through to lower level. On the contrary, bottom up approach starts with lower level modules and proceeds through higher levels to the top-level module.

3.2 Top Down Design

This approach starts by identifying major components of the system decomposing them into their own subordinate level components and interacting until the desired level of detail is achieved. Top down design methods often result in some form of stepwise refinement, starting from an abstract design, in each step, the design is refined to a more concrete level until we reach a level where no more refinement is required and the design can be implemented directly. This approach is explained by taking an example of “Library Information System” depicted in figures 6. 1, 6.2 and 6.3 below:

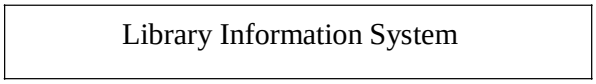


Figure 6.1: The top (root) of software system

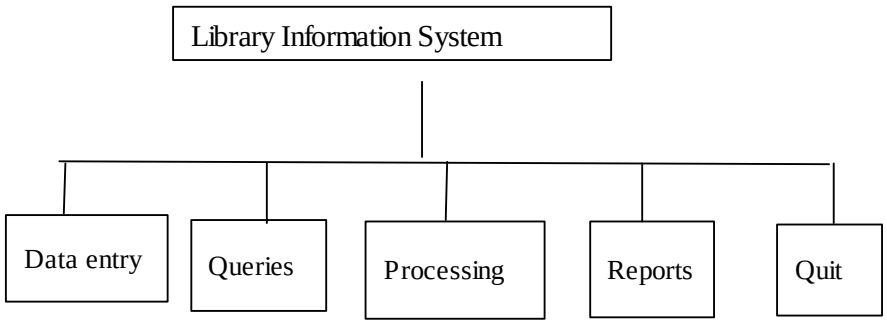


Figure 6.2: Further Decomposition of the "top" of software system

Now, we can move further down and divide the system even further as shown in figure 6.3.

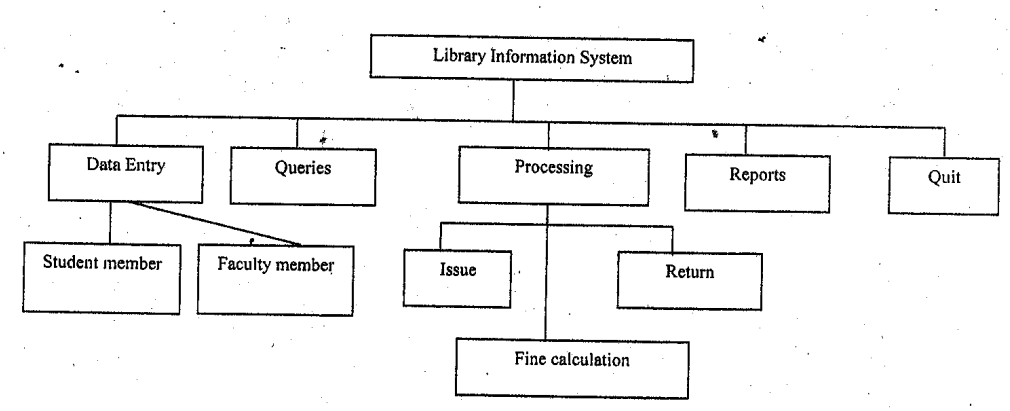


Figure 6.3: Hierarchy Chart of Library Information system

This iterative process can go on till we have reached a complete software system. A complete software system is a system that has been coded completely using any front-end tool (ex Java, Visual Basic, VC+ +, Power Builder etc).

Top down design strategies work very well for system that is made from tile scratch. We can always start from the main menu and proceed down the hierarchy designing data entry modules, queries modules, etc.

3.3 Bottom Up Down

In bottom up strategy, we start from the bottom and move upwards towards the top of the software.

This approach leads to a style of design where we decide tile process of combining modules to provide larger ones, to combine these to provide

even larger ones and so on till we arrive at one big module. That is, the whole of the desired program.

This method has one weakness. We need to use a lot of intuition to decide the functionality that is to be provided by the module. If a system is to be built from existing system, this approach is more suitable as it starts from some existing modules.

4.0 CONCLUSION

In this unit, we have learned about design principles as well as the top down and bottom up designs.

5.0 SUMMARY

What you have learned in this unit concerns design principles and requirements of a good design.

SELF ASSESSMENT EXERCISE

1. What do you understand by abstraction?
2. What policy is adopted in problem partitioning?

6.0 TUTOR-MARKED ASSIGNMENT

Describe two strategies that help important design principles.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 5 STRUCTURAL DESIGN

CONTENTS

1.0 INTRODUCTION

Structure charts represent design graphically. In this unit you will learn the process of drawing structure charts. Modularization is equally considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

learn the process of drawing a structure chart
learn the goals of design
differentiate between five types of coupling and apply then in programs
differentiate between seven types of cohesion apply them in program.

3.0 MAIN CONTENT

3.1 Structure Charts

A structure chart depicts the modular organization of an information system. The organization is hierarchical. A structure chart graphically shows the way the components of a program or a system are related. The relationships are shown in terms of parameter passing mechanisms applied and the basic structured programming operations namely sequence, selection and repetition. A structure chart depicts the division of a system into programs along with their internal structure. However, the internal structure of those programs which are written in third and fourth generation languages can be depicted.
A structure chart depicts various modules across different levels of the hierarchical organisation. Always, it has one coordinating module at the

top. The modules at the next level are called by the coordinating module. If a menu based system is concerned, the main menu may be considered as the coordinating module and the options in it may be considered as subordinate modules. Even, the calling mechanism is hierarchical. The coordinating module at the top calls the modules at the next level and the modules at this level call the modules at the next level to them. A module calls a subordinate module whenever there is a need for the operation to be performed by the subordinate module. Now, the following question arises: What about those modules at the lowest level? Whom do they call, as there are no modules after that level? The answer is: Modules at lower level perform various tasks. They don't call any other module.

Consider Figure 6.4. It depicts a structure chart. *System* is the top module. Its subordinate modules are *Get X* and *Make Y*. The subordinate modules of *Get X* are *Get* Wand *Make X*. There are no subordinate modules for *Make Y*. It is very important that the function of a module can be easily grasped from its name. So, naming of the modules is critical to understand the system as a whole. Since a module should not perform more than one task, there will be no use of *and* in the name of a module as it means that the module is performing multiple tasks. But, it is common to find modules which perform multiple tasks and this can be easily realized from the conjunctions used in their names. In the case of such modules, it is advisable to divide them into multiple modules with each module performing a single task. These modules can be executed from left to right.

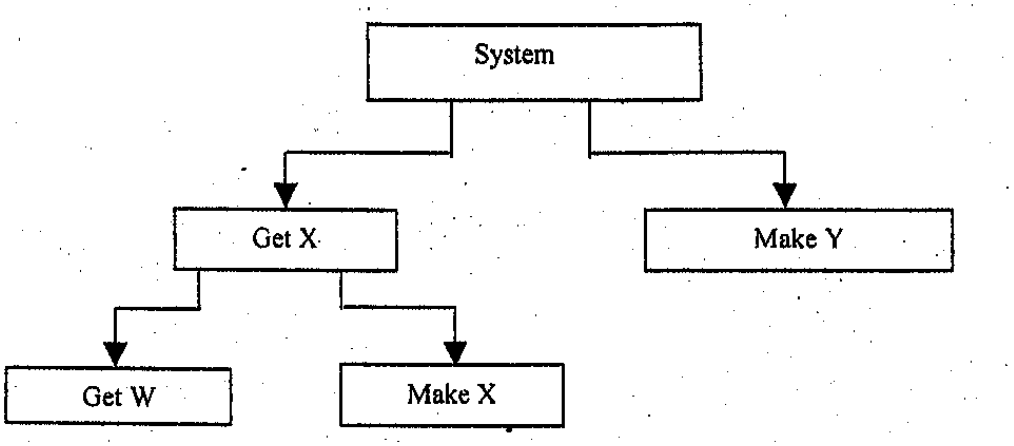


Figure 6.4: Hierarchy of a Structure Chart

The communication between various modules in a Structure chart takes place by passing the requisite data items as parameters. Data is represented as data couples and flags. A *data couple* is a symbol which consists of an empty circle with an arrow coming out of it. The direction of the arrow indicates the direction of data communication. A control

flag is indicated by using the same symbol as that of a data couple except that the circle is filled. A control flag gives information about the data being communicated. It may indicate EOF etc. Figure 6.5 shows tile symbols for data couples and Figure 6.6 show the symbols for control flag.

Figure 6.5: Data couples

Figure 6.6: Control Flag

A module is indicated by a rectangle. The name of the module may be indicated with in the module. Figure 6.7 shows the symbol for module.

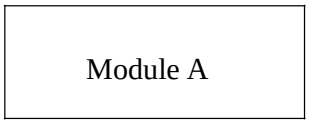


Figure 6.7: Symbol for Module

Figure 6.8 depicts a set of superordinate and subordinate modules. Here A is superordinate module and B is subordinate module. So, A will call B whenever necessary.

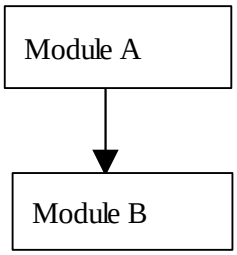


Figure 6.8: Symbols for superordinate module A and subordinate module B.

Figure 6.9 depicts a total of three modules namely A, Band C. A is the superordinate module and B, C are subordinate modules. The curved arrow over the two communication lines connecting module A to Band C indicates that Band C are repeatedly called by A.

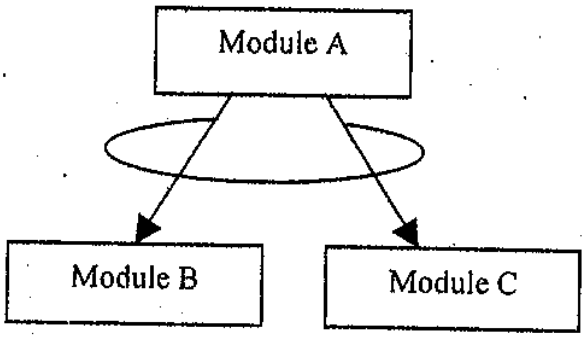


Figure 6.9: Repeated calls of Module B and C by Module A

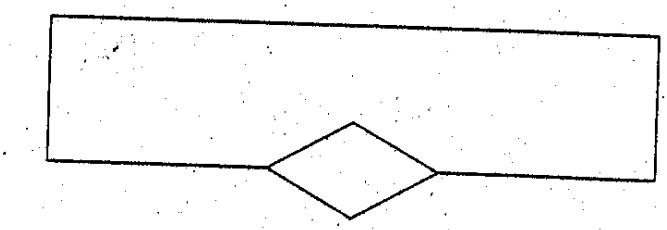


Figure 6.10: Subordinate modules are called on condition

Figure 6.10 depicts the diamond symbol. It indicates the subordinate module to be called on the result of the execution of a conditional statement. Though, there can be a number of subordinate modules for a superordinate module, not all of them are called when a diamond symbol exists.

Figure 6.11 depicts a module A flanked by two vertical bars. Presence of these bars indicates that the module is predefined. It is analogous to predefined functions or built in functions (of course, not always). These modules exist at the bottom level of a Structure chart.

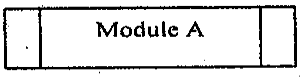


Figure 6.11: Predefined module A

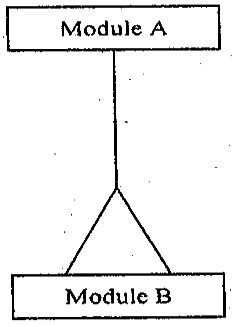


Figure 6.12: Module B is embedded in Module A.

Figure 6.12 depicts the *hat* symbol for an embedded module. Though, B is logically shown separately from A, since A and B are connected by the embedded module symbol *hat,* it means that B is in fact a part of A and physically the module does not exist separately. This is often done due to the size of such modules which is very smaller and don't fit to be called a separate module. In such cases, they become part of superordinate module.

Consider the structure chart of Figure 6.13. The system module calls Get Marks A module. This module in turn calls Read Marks A .This module sends tile requisite marks to Get Marks A. Then, Get Marks A calls Validate Marks A and also sends Marks A to it for validation. Validated marks A are sent back from Validate Marks A to Get Marks A. The process repeats again in the case of Marks B also. After obtaining validated marks in A and B, the system module calls Make Result R to compute the result. It sends marks in A and B to it. Make Result R sends tile result R to system. Finally, the system module calls Put Result R module and passes the result to it.

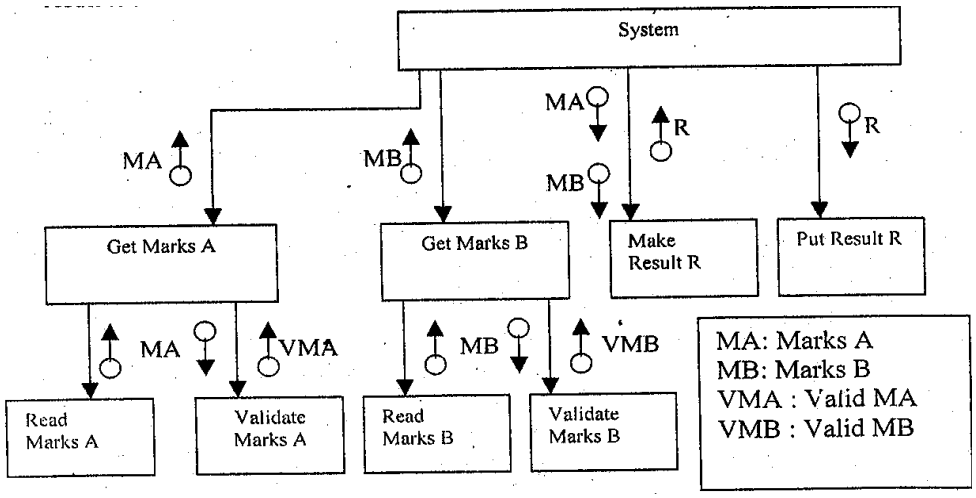


Figure 6.13: Reading a Structure chart

3.2 Modularity

According to C. Mayers, *Modularity* is a single attribute of software that allows a program to be intellectually manageable". It increases the design clarity that results in easy implementation, testing, debugging, documentation and maintenance of software product. Modularity means "decomposing a system into smaller components that can be coded separately". Modularity does not mean simply chopping off system into smaller components but certain concepts like coupling and cohesion needs to be followed while breaking a system into different modules.

3.3 Goals of Design

If there is a system which can be read easily, code easily and maintain easily, then we can come to a conclusion that the design is fine. Any design which achieves the goals given below can be termed as good design:

1. The design of the system should be module based. It means there are modules which together make up the system and the organization of these modules is hierarchical.

2. Each module controls the functions of a suitable number of subordinate modules at the next hierarchical level.

3. One of the important features of good design is that the modules, which make up the system don't communicate intensively. The communication should be kept at minimum level. The reason for this imposition is that modules should be independent of each other to the maximum extent, possible. Independence means, "one module's functionality should .not be dependent on the internal functions of other module".

4. The size of module should be appropriate as required for the features it should possess like being relatively independent of other modules etc. Basically, no specific size of range of size can be defined on modules though it is done occasionally. The size varies from module to module and from project to project.

5. A module should not be assigned the duty of performing more than one function.

6. The coding of modules should be generic. It enables the system to use the module as frequently as possible.

Based on the above listed goals, a set of guidelines for good design can be arrived at. They are given below:

A system should be divided into as many relatively independent modules as possible. This is known *as factoring.*

A superordinate module should control not more than seven subordinate modules. Of course, this guideline is not strict and varies from system to system.

The dependency levels between modules should be minimum. This automatically leads to the design of modules, which don't communicate, frequently with each other. Also, the communication between modules should be through parameters. Of course, Boolean variables or flags can be used for the purpose of communication. This is called *coupling.*

Usually, a module if of not more than 100 lines. It may be a minimum of 50 lines. But, these sizes are not to be strictly followed and they may vary from system to system It is notable here that lesser tile lines of code, easier to read.

A module should not perform more than one function. There should be no time in the code of the module, which is concerned with a function that is not the objective of that particular module. One easy check for this conformance is that the module' function should be describable easily in a few words. This is called *cohesion.*

Modules at the lower level of tile design are called by more than one superordinate module. It means that multiple superordinate modules use most of tile modules at the lower level.

3.4 Coupling

The dependency levels between modules should be minimum. This automatically leads to the design of modules that don't communicate frequently with each other. Also, the communication between modules should be through parameters. Of course, Boolean variables or flags can be used for the purpose of communication. This is called *coupling.*

The coupling between the modules should be minimum. The reason for stressing the need for minimum dependence between modules is that, if a module-l is largely dependent on another module-2, then, any error in module-2 will affect the functionality of module-l. This is the case of two modules that are largely dependent on each other. But, in the case of multiple modules being largely dependent among themselves, the consequences of errors in one or more modules will be drastic. The other problem with the dependency of one module on another module is related to maintenance. If a programmer has to change the functionality of a module then he should also make necessary changes to the internals of the modules on which the module in question is largely dependent. It automatically leads to the disturbance of the entire system. Such modular design usually leads to the, need for development of the system from the scratch which is going to have significant implications in terms of efforts to be put, amount to be spent etc. If modules are independent to the extent possible then it will become easy for the programmers to

make changes in a particular module with out making any chaJ1ges in other modules. Also, it leads to a greater reuse of the modules in multiple projects wherever the functionality of the module is needed. Though it is desirable, it is highly possible to minimize coupling among the modules.

There are five types of coupling. They are explained below:

Data Coupling: In this type of coupling, the communication between the modules is through passing of data as parameters. The other alternative in this type of coupling is the usage of flags. So, one module will not be and need not be aware of the internal structure of the module with which it is communicating.

Consider the Figure 6.14. **Prepare the salary of employee** is the superordinate module. **Calculate Salary** is the subordinate module. It is coupled with Calculate Salary. But, **Prepare the salary of employee need** not be aware of the internal structure of **Calculate Salary**. **Calculate Salary** needs to know the data being passed to it for doing the requisite task and the data that has to be returned by it to the superordinate module.

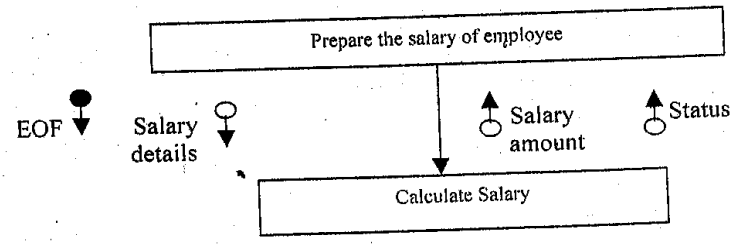


Figure 6.14: Example of Data Coupling

Stamp Coupling: The mechanism of communication in Stamp Coupling is achieved by passing data structures. Alternatively, records consisting of requisite data are sent. The problem with this type of coupling is that any changes in the data structure will lead to a chain reaction and all the modules that use this data structure have to be change. Sending data as stated in the technique of data coupling is ideal. Stamp coupling increases the dependency levels among the modules. All the modules, which are using the same data structure, should be aware of the internal functions of each other. This is required to avoid errors due to the usage of tile same data structure. Since the same data structure is being used, the entire data is passed to the subordinate module, which leads to redundant increased communication and more scope for data corruption.

Figure 6.15 demonstrates Stamp coupling. Obviously, the entire **Employee record** will contain more data than data required by the **Calculate Total Salary** module. The process **Format Payslip** then uses

data structure **Employee record**. Once again **Employee record** contains too much data-for this process. It would be better for both superordinate, subordinate modules and for the system as a whole if only the relevant data elements were passed instead of the entire record.

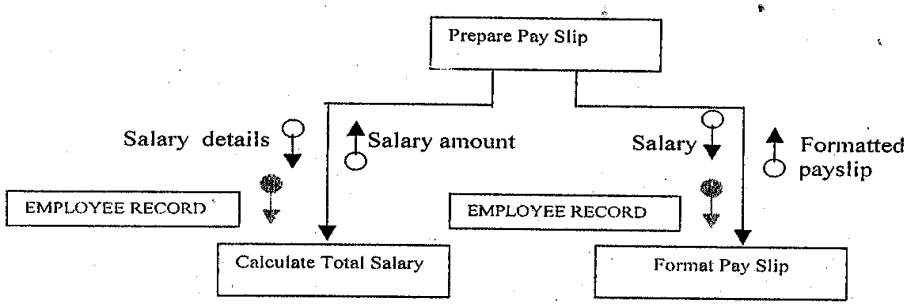


Figure 6.15: Example of Stamp Coupling

Control Coupling: In this technique of coupling, the superordinate module communicates with subordinate module by passing control information. The control information conveys the functions to the subordinate module that are to be performed by it. In this type of coupling, interdependence between the superordinate and subordinate modules is high as the superordinate module should definitely know the internal functions of the subordinate module to invoke it for a particular task. Figure 6.16 depicts an example of Control coupling. The signal that control information is being passed is that the label of the flag starts with the verb Prepare Payslip. It is to be noted that, in some cases, control information may be passed from the subordinate module to superordinate module. But, this rarely occurs.

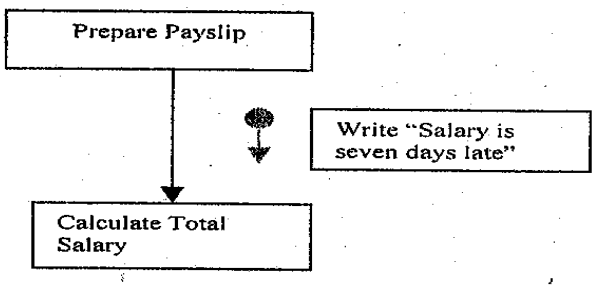


Figure 6.16: Example of Control Coupling

Common Coupling: In this technique of coupling, global data areas are used by the multiple modules. Usage of global variables is permitted in most of tile High level languages. A variable can be declared at appropriate level so that iris treated as global variable. All modules which use this data area will be accessing the data in that area that is present there at that point of time. If any module performs invalid operation on the data in the global data area then the data area holds the resultant wrong value. But this wrong value will be used by the module

which subsequently accesses this global \ data area resulting in further invalid processing. In this type of coupling, interdependence among the modules is very high as they are sharing the data area and any wrong doing by any of the modules on this global area is going to impact the processing of all the subsequent modules which use the data in this global data area.

Content Coupling: This is the technique of coupling which has to be used when none .of the above are possible. The major drawback of this technique is that one module can access the data inside another module and alter it. Also it is possible to change the code of one module by another module. This is the technique of coupling in which independence among the modules is not even slightly visible. Fortunately, most of the High level languages don't support content coupling.

All the coupling techniques that are discussed above rate from top to bottom in terms of priority. In other words, data coupling should be most sought after technique of coupling followed by stamp coupling and then control coupling followed by common coupling and lastly content coupling.

3.5 Cohesion

Cohesion reflects the degree to which a module conforms itself to the performance of a single task. A simple way to check if a module is cohesive or not, is to examine each instruction in it. If every instruction is related to the performance of a single task, then the module is said to be cohesive. Modules should be highly cohesive. Two objectives can be achieved if we strive to make a module cohesive. First is that the module will perform single task. It leads to a larger degree of portability and we can directly plug in the module in an application which requires the performance of this task. The second is that module will be loosely coupled. Since tile module is performing the single task, it will accept the data from a superordinate module, does the requisite function and returns the results. So, there is no need or minimum need to know the internal function of any other module.

There are seven types of cohesion. They are explained below:

Functional Cohesion: A module is functionally cohesive if every instruction in the module is related to a single task. One easy way to know whether the module is functionally cohesive or not is to examine its name. The name of the module will usually indicate the task that is performed by it. For example, Print Grade Cards, Generate payslips etc. are names of modules that perform a single function.

Sequential Cohesion: In this type of cohesion, all instructions in the module are related to each other through the data that is passed to the module. If each instruction is examined individually, it is difficult to know whether the module is performing single function or not. But, if the module is simulated and instruction wise simulation is examined, then we can conclude that the module is sequentially cohesive if each instruction's input data is the output data of the previous immediate instruction. In other words, the concept of sequential cohesion is similar to the concept of pipeline processing. So, in sequential cohesion, sequencing of instructions plays a major role in the cohesiveness of the module.

Consider the following example of sequential cohesion:

Produce purchase order.
Prepare shipping order,
Update invention
Update accounts.

Purchase order is the initial input for this set of instructions. Produce purchase order is the input to the second instruction where the shipping order is prepared. This instruction will serve as an input to the third instruction to update inventory and this will serve as input to update accounts. So, in this way, the output of first instruction has become the input for the second instruction, the output of the second instruction has become the input for the third instruction and so on.

Communicational Cohesion: This type of cohesion shares an analogy with sequential cohesion regarding the aspect that all instructions in the module are related by the data used by the module. But, at the same time, it differs from the sequential cohesion with no restriction on the sequencing pattern of the instructions. So, in communicational cohesion, the ordering of instructions is irrelevant. The most important thing is that, input data for each instruction is same.

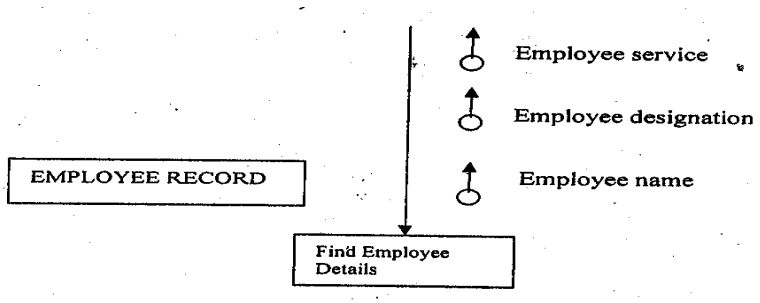


Figure 6.17: An example of a communicational Cohesion module.
Figure 6.17 depicts an example of Communicational Cohesion. **Find Employee Details** is a subordinate module which receives an Employee

record as input from the superordinate module and finds the employee's name, designation and service. To find each of name, designation and service, *Employee* record is used. So, the input for all the three instructions is same. Also, sequencing does not matter here because any of the name, designation and service details can be found at any point in the order in d the input to any of these instructions is not dependent on the output of the other instructions.

It is also possible to use two functionally cohesive modules than one communicational cohesive module. Figure 6.18 depicts two functionally cohesive nodules instead of one communicational cohesive module in Figure 6.17.

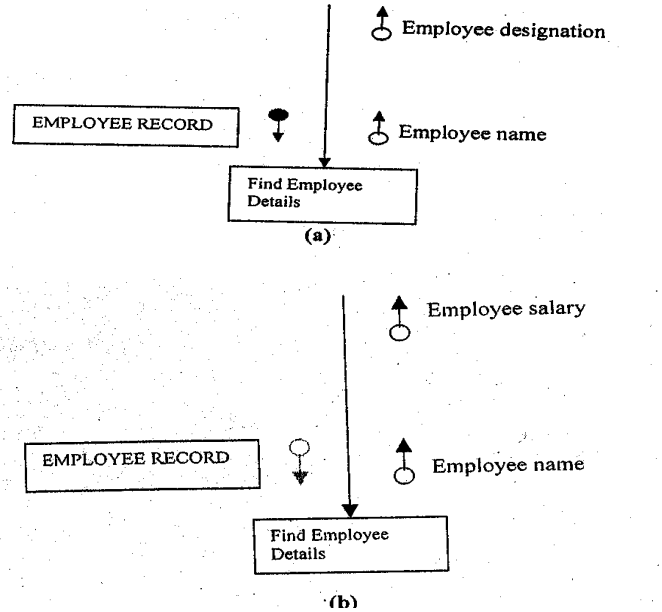


Figure 6.18 (a) & (b): An example of two functionally cohesive modules which resulted due to the spilt of one communicational cohesive module

Procedural Cohesion: Any module which is not functionally cohesive is difficult to maintain. In this type of cohesion, the sequence of instructions is important and they are related to each other by the control flow. It is possible to make a change in sequence, but it cannot be arbitrarily done. The execution of instructions in the module which is procedurally cohesive usually leads to the calls to other modules.

So, the instructions in a procedurally cohesive module are related to the instructions in other modules.

Consider the following example:

Pick the course material from MPDD,
Check the enrolment number on the Hall ticket,
Attend counselling sessions at Study Centre,
Submit assignments at Study Centre.

In the above example, instructions are not related to each other in terms of sequence. In some cases, sequence may be of importance. But, at tile same time, each instruction is separate in functionality and leads to the execution of instructions in other modules.

Temporal Cohesion: In this type of cohesion, instructions in a module are related to each other only by flow of control and are totally unrelated to their sequence. In a temporally cohesive module, execution of all the instructions may take place at a time.

Consider the following example:

Delete duplicate data from inventory file,
Reindex inventory file,
Backup of inventory file.

All these operations have nothing in common except that they are end of the day clean up activities.

Logical Cohesion: In this type of cohesion, the relation between various instructions in the module is either nil or at a bare minimum. A logically cohesive module consists of instructions in the form of sets. So, execution takes place in terms of set of instructions rather than individual instructions. But, tile superordinate module which calls the logically cohesive subordinate module will determine the set of instructions to be executed. This mechanism is handled with the help of a flag which is passed to the subordinate module by the superordinate module indicating the set of instructions to be executed.

Consider the following example:

Study in home,
Study in library,
Study in garden.

This is bad type of cohesion. It is very difficult to maintain logically cohesive modules.

Coincidental Cohesion: In this type of cohesion, there is no relationship between the instructions. This is worse type of cohesion among the discussed. The reason for placing such type of totally

unrelated instructions may be to save time from programming, to fix errors in the existing modules etc.

The priority of all the seven types of cohesion discussed moves from top to bottom. So, Functional cohesion is the best and Coincidental cohesion is the worse type of cohesions. The order of priority is as follows: Functional cohesion, Sequential cohesion, Communicational cohesion, Procedural cohesion, Temporal cohesion, Logical cohesion and Coincidental cohesion.

SELF ASSESSMENT EXERCISE

1. What are the goals of a good design?
2. Describe any two types of coupling?

4.0 CONCLUSION

The process of depicting the modular organization of a system has been explained using structure charts. The issue of degree of communication that should be present between modules through coupling and cohesion has been discussed.

5.0 SUMMARY

This unit focused on the structured design of a system.

6.0 TUTOR-MARKED ASSIGNMENT

Explain the term “modularity”.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 6 LOGICAL AND PHYSICAL DESIGN

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn about different phases of system development life cycle and the steps involved in the logical design.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

understand the logical design
describe the form design
be aware of the reports design
be able to list the guidelines for user interface design.

3.0 MAIN CONTENT

3.1 Logical Design

Is the phase of system development life cycle in which system analyst and user develops concrete understanding of the operation of the system? Figure 7.1 depicts various steps involved in the logical design. It includes the following steps:

designing forms (hard copy and computer displays) and reports, which describe how data will appear to users in system inputs and outputs;

designing interfaces and dialogues, which describe the pattern of interaction between users and software; and

designing logical databases, which describe a standard structure for the database of a system that is easy to implement in a variety of database technologies.

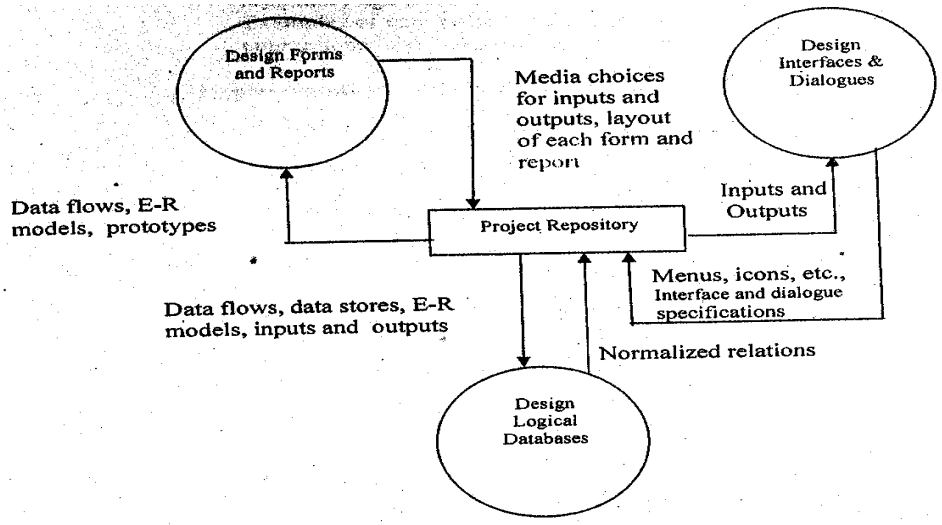


Figure 7.1: Steps in Logical Design

In logical design, all functional features of the system chosen for development in analysis are described independently of any computer platform. Logical design is tightly linked to previous system development phases, especially analysis. The three sub phases mentioned in the figure 7. 1 are not necessarily sequential. The project dictionary or CASE repository becomes an active and evolving component of system development management during logical design. The complete logical design must ensure that each logical design element is consistent with others and satisfactory to the end user.

3.2 Form Design

System inputs are designed through forms. The general principles for input design are:

1. Capture only variable data.
2. Do not capture data that can be calculated or stored in computer programs.
3. Use codes for appropriate attributes.
4. Include instructions for completing the form.
5. Data to enter should be sequenced.

Common GUI controls for Inputs

Text Box

It can allow for single or multiple lines of characters to be entered.

Radio Button

Radio buttons provide the user with an easy way to quickly identify and select a particular value from a value set. A radio button consists of a small circle and an associated textual description those responses to the value choice. Radio buttons normally appear in groups as one radio buttons per value choice.

Check Box

It consists of a square box followed by a textual description of the input field for which the user is to provide the Yes/No value.

List Box

A list box is a control that requires the user to select a data item's value from the list of possible choices. It is rectangular and contains one or more rows of possible data values. The values may appear as either a textual description or graphical representation.

Dropdown List

It is like a list box, but is intended to suggest the existence of hidden list of possible values for a data item.

Combination Box

It is also known as combo box. It combines the capabilities of a text box and list box. A combo box gives the user, the flexibility of entering a data item's value or selecting its value from a list.

Spin Box

A spin box is used to allow the user to make an input selection by using the buttons to navigate through a small set of meaningful choices.

The following steps are to be followed during the process of input design are given below:

1 Identify system inputs and review logical requirements.
2. Select appropriate GUT (Graphical User Interface) controls.
3. Design, validate and test inputs using some combination of:

i) Layout tools (e.g., hand sketches, printer/display layout chart, or CASE)
ii) Prototyping tools (e.g., spreadsheet, PC DBMS, 4GL)

4. If necessary, design the source document.

3.3 Reports Design

System outputs are designed through Reports. Outputs can be classified according to two characteristics:

i) Their distribution inside or outside the organization and the people who read and use them; and
ii) Their implementation method.

Types of outputs

i. **Internal Outputs**: Internal outputs are intended for the owners of the system and users within the organization. There are three sub-classes of internal outputs: Detailed report--Present information with little or no filtering or restrictions; Summary reports--Categorize information for managers who do not want to go through details; and

ii. **External Outputs**: These are intended for customers, suppliers, partners and regulatory agencies. They usually conclude or report on business transactions. Examples of external outputs are invoices, account statements, paycheques, course schedules, telephone bills, etc.

Implementation Methods for Outputs

The following are commonly used output formats.

i. **Tabular Output**: It presents information as rows and columns of text and numbers.
ii. **Zoned Output**: It places text and numbers into designated areas or boxes of a form or screen.
iii. **Screen Output**: It is the online display of information on a visual display device, such as CRT terminal or PC monitor.
iv. **Graphic Output**: It is the use of a picture to convey information in ways that demonstrate trends and relationships not easily seen in tabular output.

The following are various guidelines for output design:

i. Computer outputs should be simple to read and interpret.

ii. The timing of computer outputs is important.
iii. The distribution of (or access to) computer outputs must be sufficient to assist all relevant system users.
iv. The computer outputs must be acceptable to the system users who will receive them.

The steps to be followed during process of output design are given below:

i. Identify system outputs and review logical requirements.
ii. Specify physical output requirements.
iii. As necessary, design any pre-printed external forms.
iv. Design, validate and test outputs using some combination of:

a. Layout tools (e.g., hand sketches, printer/display layout chart, or CASE).
b. Prototyping tools (e.g., spreadsheet, PC DBMS, 4GL).
c. Code generating tools (e.g., report writer).

3.4 User Interface Design

User interface design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue.

The following are various guidelines for user interface design:

i) The system user should always be aware of what to do next.
ii) The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
iii) Messages, instructions or information should be displayed long enough to allow the system user to read them.
iv) Use display attributes sparingly.
v) Default values for fields and answers to be entered by the user should be specified.
vi) A user should not be allowed to proceed without correcting an error.
vii) The system user should never get an operating system message or fatal error.
viii) If the user does something that could be catastrophic, the keyboard should be locked to prevent any further input, and an instruction to call the analyst *or* technical support should be displayed.

3.5 Logical Database Design

Data modeling is used for logical database design. A conceptual model of data used in an application is obtained by using an entity relationship model (E-R model). E-R model assists in designing relational databases. A relational database consists of a collection of relations relevant for a specified application. A relation is a table which depicts an entity set. Each column in the relation corresponds to an attribute of the el1tity. Each *row* contains a member of the entity set.

Normalization is a procedure used to transform a set of relations into another set which has some desirable properties. Normalization ensures that data in the database are not unnecessarily duplicated. It also ensures that addition and deletion of entity rows (or tuples) or change of individual attribute values do not lead to accidental loss of data or errors in database.

The steps to be followed during physical design are given below:

Designing physical files and databases--describes how data will be stored and accessed in secondary computer memory and how the quality of data will be ensured.

Designing system and program structure--describes the various programs and program modules that correspond to data flow diagrams and other documentation developed in earlier phases of lifecycle.

Designing distributed processing strategies--describes how your system will make data and processing available to users on computer networks within the capabilities of existing computer networks.

Figure 7.2: depicts various steps involved in physical design

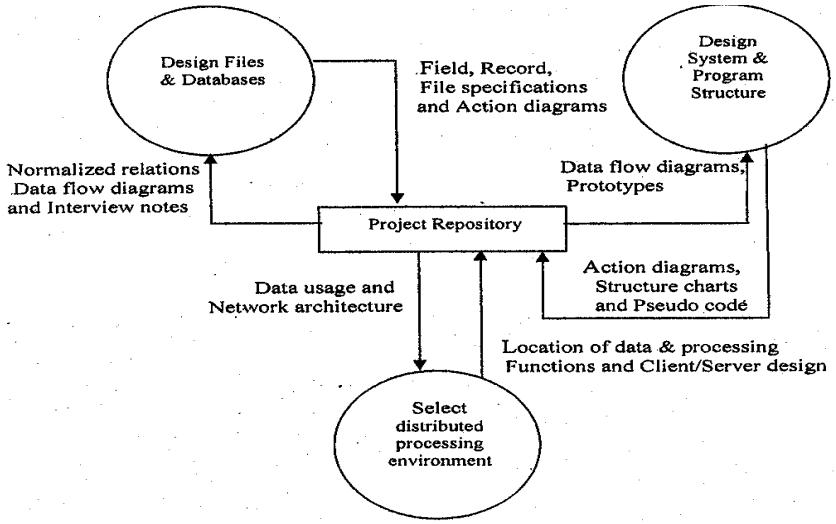


Figure 7.2: Steps in Physical Design

SELF ASSESSMENT EXERCISE

1. Describe the user interface design.
2. Explain two types of outputs.

4.0 CONCLUSION

The logical, form, reports, user interface and logical database design were considered in this unit.

5.0 SUMMARY

This unit focused on the logical and physical design.

6.0 TUTOR-MARKED ASSIGNMENT

Explain the term “modularity”.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 7 PROCESS MODELING

CONTENTS

1.0 INTRODUCTION

Process modeling, data flow diagrams, components of a data flow diagram and rules for drawing them will be considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

describe process modeling
list the categories of data flow diagrams
describe the components of a data flow diagram
declare the rules for drawing a data flow diagram.

3.0 MAIN CONTENT

3.1 Process Modeling

Process Modeling involves graphically representing the functions or processes, which capture, manipulate, store and distribute data between a system and its environment and between components within a system. A common form of a process model is data **flow diagram**. It represents the system overview.

3.2 Data Flow Diagram

A DFD can be categorized in the following forms:

Context Diagram: An overview of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system. In this diagram, a single process represents the whole system.

First Level DFD: A data flow diagram that represents a system's major processes, data flows, and data stores at a high level of detail.

Functional Decomposition Diagram: Functional decomposition is an iterative process of breaking the description of a system down into finer and finer detail which creates a set of charts in which one process on a given chart is explained in greater detail on another chart. In this diagram, sub-processes of first level DFD are explained in detail.

There is no limit on the number of levels of Data Flow Diagrams that can be drawn. It depends on the project at hand.

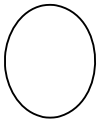
3.3 Components of a Data Flow Diagram

The following are various components of a Data Flow Diagram:

3.3.1 Process

During a process, the input data is acted upon by various instructions whose result is transformed data. The transformed data may be stored or distributed. When modeling the data processing of a system, it doesn't matter whether the process is performed manually or by a computer. People, procedures or devices can be used as processes that use or produce (transform) data.

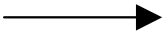
The notation (given by Yourdon) for process is



3.3.2 Data Flow

Data moves in a specific direction from a point of origin to point of destination in the form of a document, letter, telephone call or virtually any other medium. The data flow is a "packet" of data.

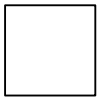
The notation (given by Yourdon) for data flow is



3.3.3 Source or Sink of Data

The origin and /or destination of data some times referred to as external entities. These external entities may be people, programs, organization or other entities that interact with the system but are outside its boundaries. The term source and sink are interchangeable with origin and destination.

The notation (given by Yourdon) for source or sink is



3.3.4 Data Store

A data store is data at rest, which may take the form of many different physical representations. They are referenced by a process in the system. The data store may reference computerized or non-computerized devices.

Notation (given by Yourdon) for data store is

3.4 Rules for Drawing a Data Flow Diagram

1. For Process

i. No process can have only outputs.
ii. No process can have only inputs.
iii A process has Ii verb phrase label.

2. For Data Store

i. Data cannot move directly from one data store to another data store. Data must be moved through a process.
ii. Data cannot move directly from an outside source to data store. Data must be moved through a process that receives data from the source and places it into the data store.
iii. Data cannot move directly to an outside sink from a data store. Data must be moved through a process.

A data store has a noun phrase label.

3. For Source/Sink

i. Data cannot move directly from a source to a sink. It must be moved by a process
ii. A source/sink has a noun phrase label

4. For Data Flow

i. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read operation before an update.
ii. A data flow cannot go directly back to tile same process it leaves. There must be at least one other process which handles the data flow, produces some other data flow and returns the original data flow to the beginning process.
iii. A data flow to a data store means update (delete or change).
iv. A data flow from a data store means retrieve or use.
v. A data flow has a noun phrase label.

SELF ASSESSMENT EXERCISE

1. List the components of a data flow diagram.
2. Mention four rules for drawing a data flow diagram.

4.0 CONCLUSION

In this unit, we have learned about process modeling. We have equally considered the components of a data flow diagram as well as the rules for drawing a data flow diagram.

5.0 SUMMARY

We considered process modeling and data flow diagrams in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

Disuses the categories of a data flow diagram.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 8 DATA MODELING

CONTENTS

1.0 INTRODUCTION

In this unit we will learn about data modeling which is also referred to as information modeling. We will also consider the entity relationship (ER) diagram as well as process specification tools.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

understand and explain data modeling describe
the entity relationship diagram describe the
three tools for specifying policies.

3.0 MAIN CONTENT

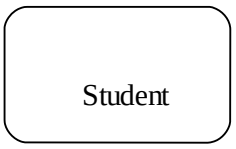
3.1 Data Modeling

It is a technique for organizing and documenting a system's data. Data modeling is sometimes called database modeling because a data model is eventually implemented as database. It is also some times called information modeling. The tool for data modeling is entity relationship diagram.

3.2 ER Diagram

It depicts data in terms of entities and relationships described by the data. Martin gives the following notations for the components of ERD.

1. **Entities**: An entity is something about which the business needs to store data. An entity is a class of persons, places, objects, events or concepts about which we need to capture and store data. An entity instance is a single occurrence of an entity. The notation is given below:



Student is the name of entity

2. **Attribute**: An attribute is a descriptive property or characteristic of an entity. Synonyms include element, property and field.

A compound attribute is one that actually consists of other attributes. It is also known as a composite attribute. An attribute "Address" is the example of compound attribute as shown in the following illustration.

3. **Relationships**: A relationship is a natural business association that exists between one or more entities. The relationship may represent an event that links the entities.

The following are some important terms related to ER diagrams:

Cardinality defines the minimum and maximum number of occurrences of one entity that may be related to a single occurrence of the other entity. Because all relationships are bidirectional, cardinality must be defined in both directions for every relationship. Figure 7.4 depicts various types of cardinality.

Degree: The degree of a relationship is the number of entities that participate in the relationship.

Recursive Relationship: A relationship that exists between different instances of the same entity is called recursive relationship. Figure 7.3 depicts recursive relationship between the instances of the *Course* entity.

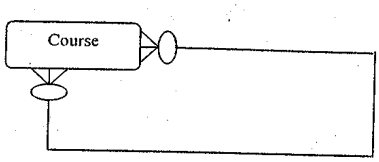


Figure 7.3: Example of Recursive relationship

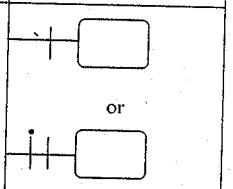
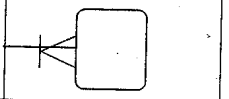
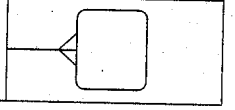
Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one (One and only one)	1	1	
Zero or one	0	1	
One or more	1	Many (>1)	
Zero, one or more	0	Many (>1)	
More than one	>1	>1	

Figure 7.4: Different types of cardinality

3.3 Process Specification Tools

Processes in Data Flow Diagrams represent required tasks that are performed by the system. These tasks are performed in accordance with business policies and procedures.

Policy

A policy is a set of rules that govern some task or function in the business. Policies consist of rules that can often be translated into computer programs. Systems Analyst along with representatives from the policy making organization can accurately convey those rules to the computer programmer for programming purposes.

Procedures

Procedures put the policies into action. Policies are implemented by procedures. Procedures represent the executable instructions in a computer program.

There are tools with the help of whom specification for policies can be created. They are Decision Table, Decision Tree and Software Engineering notations.

3.3.1 Decision Tables

Decision Table is very useful for specifying complex policies and decision-making rules. Figure 7.5 depicts a Decision table.

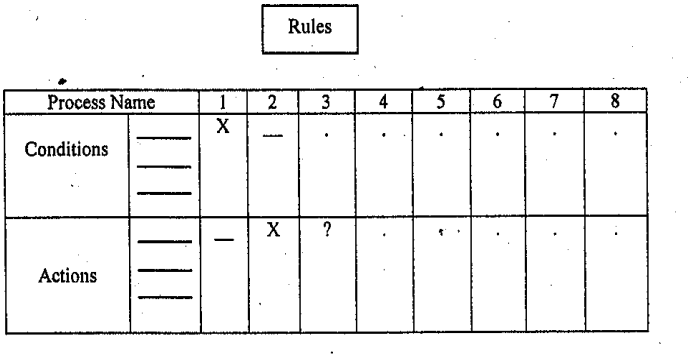
The following are various components of a Decision table:

Condition Stubs: This portion of table describes the conditions or factors that will affect the decision or policy making of the organisation.

Action Stubs: This portion describes the possible policy actions or decisions in the form of statements.

Rule: Rules describe which actions are to be taken under a specific combination of conditions.

Decision tables use a standard format and handle combinations of conditions in a very concise manner. Decision table also provides technique for identifying policy incompleteness and contradictions.



X – Action (condition is true)
_ - Condition is irrelevant for this rule
? – Unknown rule

Figure 7.5: An example Decision Table

3.3.2 Decision Trees

Decision tree is a diagram that represents conditions and actions sequentially, and thus shows which conditions to consider first, and so on. It is also a method of showing the relationship each condition and permissible subsequent actions. The diagram resembles branches on a tree.

The root of the tree is the starting point of the decision sequence. The particular branch to be followed depends on the conditions that exist and decision that will be made. Progression from the 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. Figure 7.6 depicts a Decision tree.

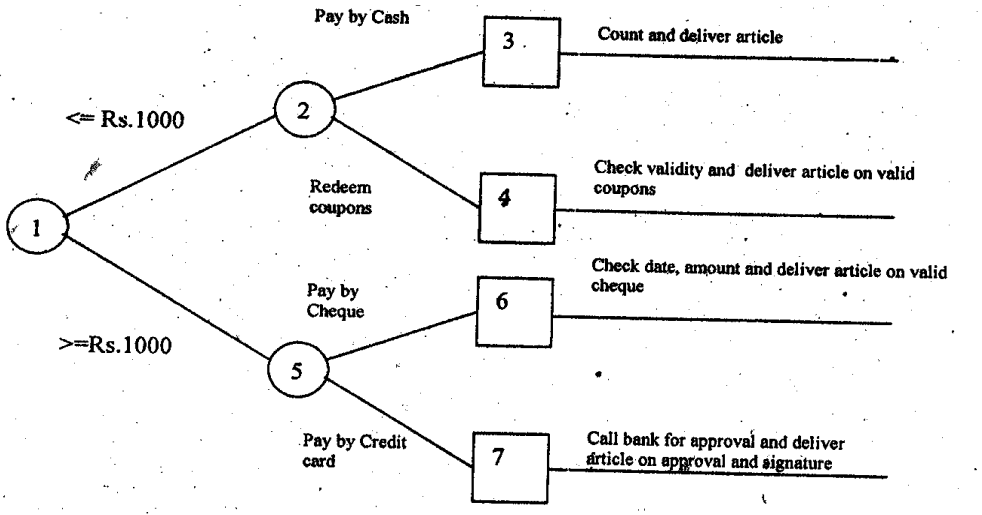


Figure 7.6: An Example Decision tree for a Customer Bill Payment System

3.2.3 Structured English Notation

It is a tool for describing process. There are three valid constructs in this notation.

They are:

1. A sequence of single declarative statements.
2. The selection of one or more declarative statements based on a decision, e.g., if-then-else, switch, case.
3. The repetition of one or more declarative statements, e.g., looping constructs such as do-while, for-do

The following are the guidelines usage of Structured English notation:

1. Avoid computer programming language verbs such a move, open or close.
2. The statement used in the Structured English Notation should always specify the formula to be used.

Structured English notation is based on the principles of structured programming. Process specification logic consists or a combination of sequences of one or more imperative sentences with decision and repetition constructs.
Consider the following:

1. An imperative sentence usually consists of an imperative verb followed by the contents of one or more data stores on which the verb operate;
   For example, add PERSONS-SALARY TO TOTAL SALARY
2. In imperative sentences, verbs such as "process", "handle" or "operate" should not be used.
3. Verbs should define precise activities such as "add" or compute average etc.
4. Adjectives that have no precise meaning such as "some" or "few" should also not be used in imperative sentences, because they cannot be used later to develop programs.
5. Boolean and arithmetic operations can be used in imperative statements. Table 7.1 lists the arithmetic and Boolean operators.

Table 7.1: Arithmetic and Boolean Operators

Arithmetic	Boolean
Multiply (*)	AND
Divide (/)	OR
Add (+)	NOT
Substract (-)	Greater Than (>)
Exponentiate (**)	Less Than (<)
	Less Than or Equal To (<=)
	Greater Than or Equal to (>=)
	Equals to (=)
	Not equal to (≠)

6. Structured English logic:

   Structured English uses certain keywords to group imperative sentences and define decision branches and iterations. These keywords are:

(BEGIN, END), (REPEAT, UNTIL), (IF, THEN, ELSE), (DO; WHILE), FOR, CASE, etc.

7. Grouping imperative sentences:

1) Sequence Construct

A sequence of imperative statements can be grouped by enclosing them with BEGIN and END keywords

2) Decision (Selection)

a) A structure, which allows a choice between two groups of imperative sentences. The key words IF, THEN and ELSE are used in this structure. If a condition is 'true', then group I sentences are executed. If it is false, then group 2 sentences are executed.

b) A structure which allows a choice between any numbers of groups of imperative sentences. The keywords CASE and OF are used in this structure. The value of a variable is first computed. The group of sentences that are selected for execution depends on that value.

3) Repetition

This structure shows two ways of specifying iterations in structured English.

a) One way is to use the WHILE...DO structure. Here, the condition is tested before a set of sentences is processed.

Alternative to WHILE….. DO is FOR structure.

b) REPEAT.....UNTIL structure. Here, a group of sentences is executed first then the condition is tested. So, in this structure, the groups of sentences are executed at least once.

Table 7.2 depicts the criteria to be used for deciding the notation among Structured English. Decision Tables and Decision Trees. Table 7.3 depicts the criteria to be used for deciding the notation among Decision Tables and Decision Trees.

Table 7.2: Criteria for deciding the notation to be used

Criteria	Structured English	Decision Tables	Decision Trees
Determining condition & actions	2	3	1
Transforming condition & actions into sequence	1	2	1
Checking consistency & completeness	2	1	1

Table 7.3: Criteria for deciding the notation to be used between Decision tables and Decision trees

Criteria	Decision Tables	Decision Trees
Portraying complex logic	Best	Worst
Portraying simple problems	Worst	Best
Making decisions	Worst	Best
More compact	Best	Worst
Easier to manipulate	Best	Worst

3.4 Data Dictionary

A Data Dictionary consists of data about data. The major elements of data dictionary are data flows, data stores and processes. The data dictionary stores details and descriptions of these elements. It does not consist of actual data in the database. But, DBMS cannot access data in database with out accessing data dictionary.

If analysts want to know tile other names by which a data item is referenced in the system or where it is used in the system, they should be able to find the answers in properly developed data dictionary. Data dictionaries are hidden from users so that data in it not tampered.

Analysts use data dictionaries for tile following reasons:

1. To manage the detail 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 changes that should be made to the system.
5. To locate errors and omission~ in the system.

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-elements are recorded in data dictionary at the fundamental data level. Each item is identified by a data name, description, alias and length and has specific values that are permissible for it in the system.

A data structure is a set of data items that are related to one another and then collectively describe a component in the system. Data is arranged in accordance with one of the relationships namely sequence, selection, iteration and optional relationship.

SELF ASSESSMENT EXERCISE

1. What is the ER Diagram?
2. Describe a recursive relationship with the aid of a diagram.

4.0 CONCLUSION

In this unit, we have learned about data modeling. We have also considered the entity relationship diagram as well as process specification tools.

5.0 SUMMARY

Data modeling, entity relationship diagram and process specification tools were considered in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

Define the term “Data modeling”.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 9 FORMS AND REPORTS DESIGN I

CONTENTS

1.0 INTRODUCTION

This unit deals with the interface of software with users. Usually the interface is through forms and reports associated with the system. In this unit, we will consider the differences between forms and reports, deliverables and design specifications.

2.0 OBJECTIVES

After going through this unit, you should be able to:

define forms and reports and their importance
distinguish between forms and reports
explain the process of designing forms and reports
describe design specifications.

3.0 MAIN CONTENT

3.1 Forms

Like a form on paper that is used to fill out information with a pen or pencil, a Form in computer terminology identifies the data we want to collect. It also allows us to enter data into the database, display it for review and also print it for distribution. However, an electronic form has several important advantages over standard paper forms. These have the

advantage of using a computer database and are more versatile and powerful that paper forms.

Examples of forms are Business forms, Electronic spread sheet, A TM transaction layout, etc.

Figure 8.2: shows a simple form that is used to collect employee details.

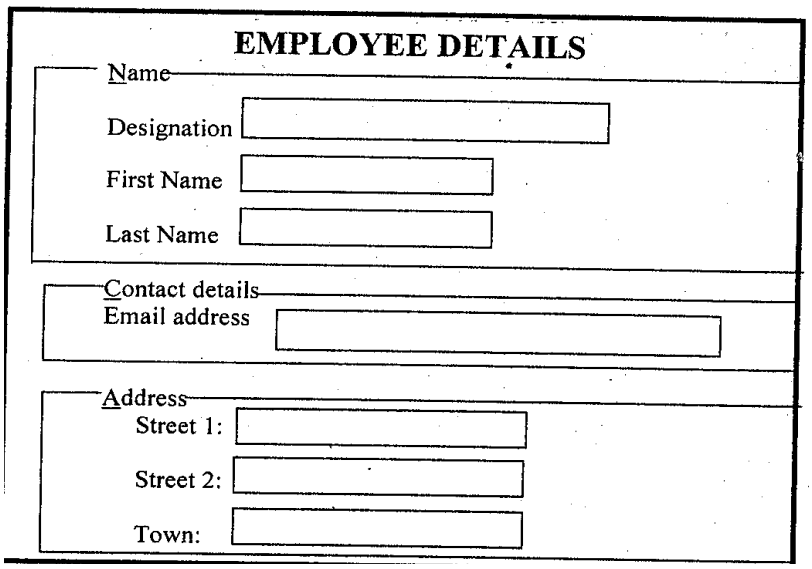


Figure 8.2: A Simple Form

Importance of Forms

The following are various advantages of Forms:

A form provides an easy way to view data.
Using forms, data can be entered easily. This saves time and prevents typographical errors.
Forms present data in an attractive format with special fonts and other graphical effects such as colour and shading.
Forms offer the most convenient layout for entering, changing and viewing records present in the database.
An entry field in a form can present a list of valid values from which users can pick to fill out the field easily.

3.2 Reports

Analysing and presenting data are just as important as entering and sorting these out. Computer systems use reporting and query applications to retrieve the data that are available in the database and present it in a way that provides useful information, drives decision-

making and supports business projects. A report presents data as meaningful information, which can be used and distributed.

A report is the information that is organized and formatted to fit the required specification. It is a passive document that contains only predefined data and is used solely for viewing and reading. Reports can be printed on paper, or these may be transferred to a computer file, a visual display screen, etc. Reports are the most visible component of a working information system and hence they often form the basis for the users and management's final assessment of the systems value.

Examples of reports are: invoices, weekly sales summaries, mailing labels, pie chart, etc.

Figure 8.3 shows a simple report that displays the residence telephone numbers of all employees in the nrl1anization.

EMPLOYEE RESIDENCE PHONE LIST

S.No	LAST NAME	FIRST NAME	DESIGNATION	PHONE NUMBER
1	Verma	Ajay	Regional Manager	6522081
2	Gupta	Vinay	Branch Manager	6478017
3	Michael	Nancy	H.R Manager	6152430
4	Singh	Amar	Sales Executive	5769081

Figure 8.3: A Simple Report

Importance of Reports

The following are various advantages of Reports:

We can organize and present data in groups.
We can calculate running totals, group totals, grand totals, percentage of totals, etc.
Within the body of Reports, we can include sub-forms, sub-reports and graphs.
We can present data in an attractive format with pictures, special fonts and lines.
We can create a design for a report and save it so that we can use it over and over again.

3.3 Differences between Forms and Reports

The following are some differences between Forms and Reports:

Forms can be used for both input and output. Reports, on the other hand, are used for output, i.e., to convey information on a collection of items.

Typically, forms contain data from only one record, or are at least based on one record such as data about one student, one customer, etc. A report, on the other hand is only for reading and viewing. So, it often contains data about multiple unrelated records in a computer file or database.

Although we can also print forms and datasheets, reports give more control over how data are displayed and show greater flexibility in presenting summary information.

3.4 Process of Designing Forms and Reports

Good quality business processes deliver the right information to the right people in the right format and at the right time. The design of forms and reports concentrates on this goal.

Designing of forms and reports is a user-focused activity that typically follows a prototyping approach. Before designing a form or a report, we should have a clear idea so as to what is the aim of the form or report and what information is to be collected from the user.

There are some useful questions related to the creation of all forms and reports, such as “who, what; when, where and how” which must be answered in order to design effective forms and reports.

WHO Understanding who the actual users are, their skills and abilities, their education level, business background, etc., will greatly enhance the ability to create effective design.

WHAT We need to have a clear understanding of what is the purpose of the form or report and what tasks will the users are performing and what information is required so as to successfully complete the given task.

WHEN Knowing when exactly the form or report is needed and used will help to set up time limits so that the form or report can be made available to the users within that time frame.

WHERE Where will the users be using this form or report (i.e., will the users have access to on-line systems or will they be using them in the field)?

HOW How many people will be using this form or report, i.e., if the form or report *is* to be used by a single person, then it will be simple In design but if a large number of people are going to use it; then the design will have to go through a more extensive requirements collection and usability assessment process.

After having answered all the above questions, we would have collected all the initial requirements. The next step is to refine this information into an initial prototype. Structuring and refining the requirements are completed without interacting with the end users, although we may need to occasionally contact users in order to clarify some issues that might have been overlooked during analysis.

Once the initial prototype is ready, we should ask the users to review and evaluate the prototype. After the review, the design may be accepted by the users. Or at times the users may ask for certain changes to be made. In case changes are to be made then the construction-evaluation-refinement cycle will have to be repeated until the design is accepted.

The next step in the Design process is to Design, Validate and Test the outputs using some combination of the following tools:

a) Layout tools (Ex.: Hand sketches, printer/display layout charts or CASE)

b) Prototyping tools (Ex.: Spreadsheet; PC, DBMS, 4GL)

c) Code generating tools (Ex.: Report writer)

The initial prototype may be constructed in numerous environments. For example, a CASE too or the standard development tools that are used within the organization be used. Usually, initial prototypes are mock screens that can be produced using word processor, computer graphics design package, or electronic spreadsheet. Mock screens are not the working modules or systems.

Tools for designing forms and reports are rapidly evolving and nowadays online graphical tools for designing forms and reports are very much in use in most professional development organizations.

3.5 Deliverables and Outcomes

Each phase in the System Development Life Cycle (SDLC) helps in the construction of the system. As we move from one phase to another, each phase produces some deliverables (measurable result or output of a process) that will be used in the later phases or activity. While designing forms and reports, design specifications are the major deliverables and these serve as inputs to the system implementation phase.

3.6 Design Specifications

Design specifications which are major deliverables while designing forms and reports have the following three sections:

Narrative overview
Sample design
Testing and usability assessment.

3.6.1 Narrative Overview

This contains the general overview of the characteristics of actual users of the form or report, task, the system that will be used and the environment factors in which the form or report will be used. The main purpose of Narrative overview is to provide information in detail to the people who will develop the final form or report, regarding the main aim of the form, who the actual users of the form will be, and how it will be used, so that they can make appropriate implementation decisions.

3.6.2 Sample Design

The sample design of the form may be hand-drawn using a coding sheet or it may be developed using CASE or standard development tools. If the sample design is done using actual development tools, then the form can be thoroughly tested and assessed.

3.6.3 Testing and Usability Assessment

This section provides information required for testing and usability assessment. While testing, it is important to use realistic or reasonable data and demonstrate all controls.

SELF ASSESSMENT EXERCISE

1. What are the advantages of forms?
2. Give two typical examples of reports.

4.0 CONCLUSION

In this unit, you would have learned about forms and reports and the differences between the two. You would also have understood what the deliverables are and the design specifications.

5.0 SUMMARY

What you have learned in this unit concerns forms and reports as well as their design specifications.

6.0 TUTOR-MARKED ASSIGNMENT

Drew the systems development life cycle with logical design.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 10 FORMS AND REPORTS DESIGN II

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn of the different types of information. You will also learn about the criteria for designing a form and report.

2.0 OBJECTIVES

After going through this unit, you should be able to:

list the different types of information
specify the criteria for form design
specify the criteria for report design.

3.0 MAIN CONTENT

3.1 Types of Information

The main purpose of any form or report is to convey certain information to the user. Information can be classified according to their distribution inside or outside the organization and the people who read and use them as follows:

Internal information,
External information, and
Turnaround information.

A. Internal Information

Internal information is the information that is collected, generated or consumed within an organization. That is, this information is intended

only for the internal system owners and system users within an organization.

Internal information either supports day to day business operations or management monitoring and decision making, e.g.,. Detailed summary or exception information printed on hard copy reports for internal business use. Internal information can also consists of simple informational reports summarizing daily activities within the organization.

The following are three sub classes of internal information:

Detailed Reports: These reports present information with little or no filtering or restrictions.

Ex.: Detailed listing of all customer accounts, orders or products in inventory. The example in figure 8.4 shows a listing of all purchase orders that were generated on a particular date. The P.O. No. Indicates the product order number.

Summary Reports: These reports categorize information for managers who do not want to wade through details. The data in summary reports are categorized and summarized to indicate trends and potential problems. We call also include charts and graphs in the summary report so that it clearly summarizes trends at a glance.

Ex.: Report that summarizes the months and years total sales by product types and category

.

The example in figure 8.5 summarizes the Sales details for a given month by product type and category.

Exception Reports: These reports filter data before they are presented to the manager as information, i.e., these reports include exceptions to some conditions or standards.

Ex.: A report that identifies items, which are low in stock.

The example in figure 8.6 depicts the identification of dealers who have to pay the dues.

THE PHILIPS STORE

Products ordered on 1-1-2003

P.O. No	PRODUCT CODE	DESCRIPTION OF GOODS	QUANTITY	RATE	AMOUNT
33473	W-37	Washing Machine	2	10,000	20,000
	T-40	21'' Colour TV	4	15,000	60,000
	R-77	Refrigerator (200ltr)	5	10,000	50,000
33475	A-339	Audio Player	7	2,000	14,000
33479	W-37	Washing Machine	5	10,000	50,000
	O-677	Microwave Oven	5	15,000	75,000
	T-40	21'' Colour TV	3	15,000	45,000

Return To Summary | Close

Figure 8.4: A Detailed Report

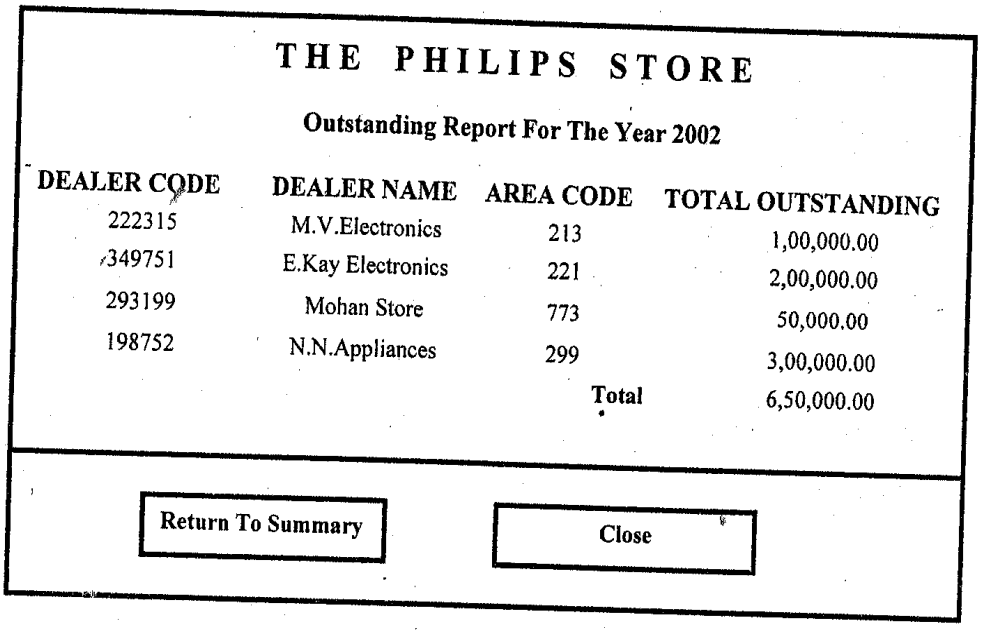
THE PHILIPS STORE

Summary Of Washing Machines Sold For The Month Of Jan 2003

PRODUCT CODE	DESCRIPTION OF GOODS	TARGETED SALE	ACTUAL SALE	VARIATION
W-37	Automatic (Top Loading)	30	25	-5
W-38	Automatic (Front Loading)	40	60	+20
W-39	Semi-Automatic (Top loading)	45	50	+15

View additional Reports | Close

Figure 8.5: A Summary Report



THE PHILIPS STORE

Outstanding Report For The Year 2002

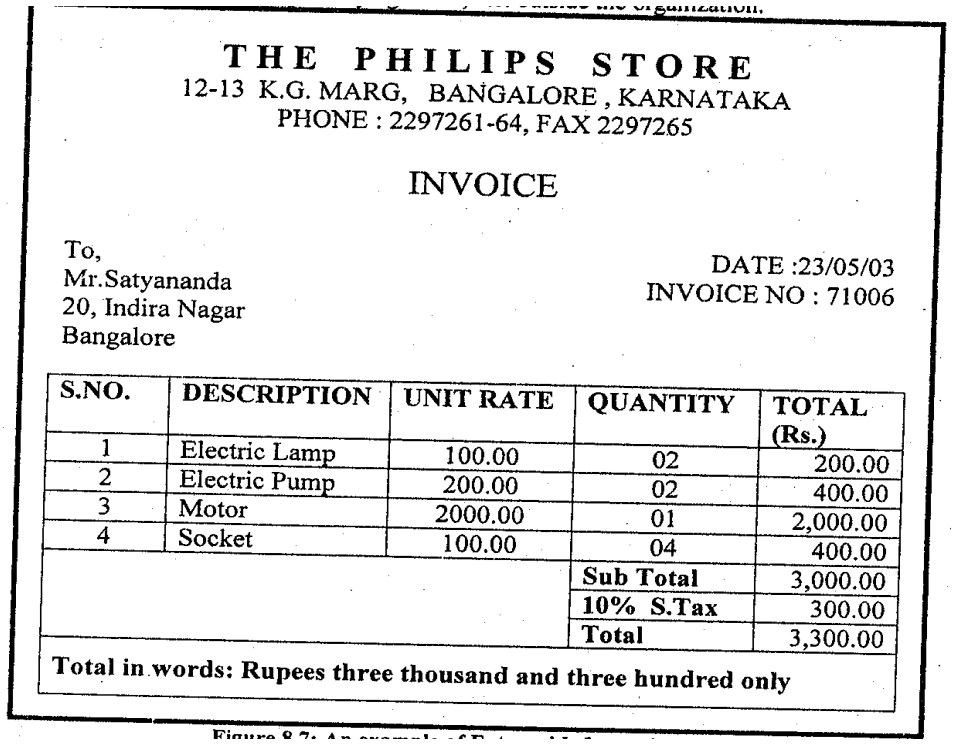
DEALER CODE	DEALER NAME	AREA CODE	TOTAL OUTSTANDING
222315	M.V.Electronics	213	1,00,000.00
349751	E.Kay Electronics	221	2,00,000.00
293199	Mohan Store	773	50,000.00
198752	N.N.Appliances	299	3,00,000.00
		Total	6,50,000.00

Return To Summary Close

Figure 8.6: Exception Report

B External Information

External information refers to the information collected from or created for customers, suppliers, competitors, regulatory, agencies, etc. outside the organization.



THE PHILIPS STORE
12-13 K.G. MARG, BANGALORE , KARNATAKA
PHONE : 2297261-64, FAX 2297265

INVOICE

To,
Mr.Satyananda
20, Indira Nagar
Bangalore

DATE :23/05/03
INVOICE NO : 71006

S.NO.	DESCRIPTION	UNIT RATE	QUANTITY	TOTAL (Rs.)
1	Electric Lamp	100.00	02	200.00
2	Electric Pump	200.00	02	400.00
3	Motor	2000.00	01	2,000.00
4	Socket	100.00	04	400.00
			Sub Total	3,000.00
			10% S.Tax	300.00
			Total	3,300.00

Total in words: Rupees three thousand and three hundred only

Figure 8.7: An example of External Information

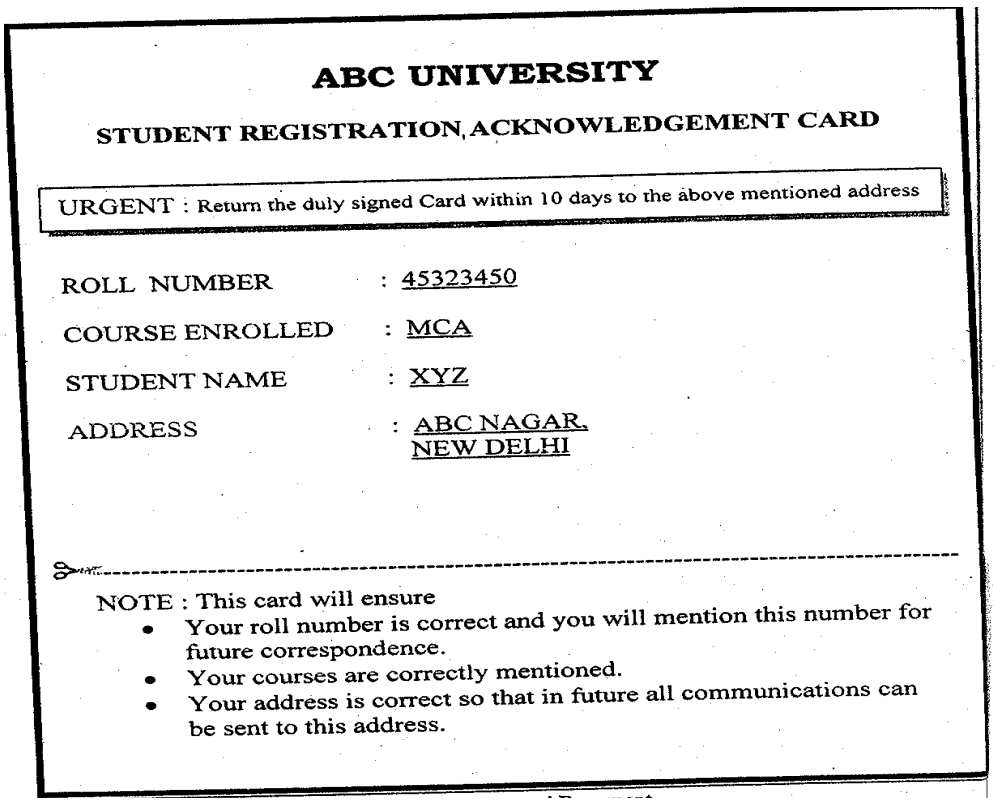
Examples of external information are: Invoices account statements, Product documentation, Purchase orders, Mailing labels, etc.

Figure 8.7 shows an invoice of a store that sells products manufactured by the Philips Company.

C. Turnaround Documents

There will be several instances where the information output is again used as input to obtain new information. The document that consists of such information is called Turnaround document. These begin as external information delivered to an external customer as an output, but ultimately return (in part or in whole) as internal information to provide new information as an input to an information system. For example, warranty card or acknowledge slip. This form has pre-printed system generated information that a customer must verify and return to the organization. The customer also adds new information (Ex.: Name, Address, etc.), which serves as input to the customer tracking system.

Figure 8.8 shows a simple acknowledgement card which is sent by a university to the student and the student in turn checks the information and sends a part of the document back to the university.



ABC UNIVERSITY

STUDENT REGISTRATION, ACKNOWLEDGEMENT CARD

URGENT : Return the duly signed Card within 10 days to the above mentioned address

ROLL NUMBER : 45323450

COURSE ENROLLED : MCA

STUDENT NAME : XYZ

ADDRESS : ABC NAGAR,
NEW DELHI

NOTE : This card will ensure

- Your roll number is correct and you will mention this number for future correspondence.
- Your courses are correctly mentioned.
- Your address is correct so that in future all communications can be sent to this address.

Figure 8.8: Turnaround Document

3.2 Criteria for Form Design

Forms should be well conceived and attractive in design. We can achieve this goal if we design a form that satisfies the following criteria:

Organization,
Consistency,
Completeness,
Flexible entry, and
Economy.

A. Organization

The different parts of a form must be arranged in a proper order with visual separation between the parts. Balancing of different information on the form should be done according to the sequence of entry, frequency of use, function and significance of that particular data. The first data available, the most important data and the data that is going to be used most frequently should always be placed in the beginning of the form. If there are groups of data of the like information, they should be placed together just as Name + Address + Phone Number Grouping of information will help *the* user to understand which section of the form they are completing.

B. Consistency

Forms designed should be internally consistent. They must also be consistent with related forms and with other forms in the organization. If the forms are consistent, then it will be easy for the users to learn how to fill them. Consistent forms reduce errors and data capture costs.

C. Completeness

The form should gather all the necessary data at the source so that there is no need to transcribe data to other forms. This reduces the major source of errors.

D. Flexible Entry

It should be possible to enter data by hand or with a typewriter. In most cases, both kinds of entries occur.

E. Economy

The total cost of design, printing, data entry, etc., must be minimized. Most of the times, it is required to increase one cost to reduce another

Usually, handling costs are much more than the cost of designing and printing Having spent more resources on design and printing often reduces the cost of data capture and keying.

3.3 Criteria for Report Design

Reports convey information from one of more computer files to the user. They perform this task satisfactorily only when they present information to the user accurately and in small portions.

Several criteria that should be considered in order to produce good reports are given below:

Relevance,
Accuracy,
Clarity,
Timeliness, and
Cost

A. Relevance

Only the information that is relevant to the purpose of the report should be present in the report. This is a selection process, i.e., all the relevant information should be included and all the irrelevant information or data should be excluded. Only required information should be printed or displayed. In on-line reports, we should use information hiding and provide methods to expand and contract levels of information details

B. Accuracy

The data that appears on the report should be accurately recorded, accurately transmitted and accurately transformed into summary data Accuracy is very important because if the data are inaccurate, then the main purpose of the report which is to provide accurate information to the user will not be accomplished. Incomplete data are also inaccurate.

C. Clarity

The information that is present on the report should be clear and understandable. The information present should be balanced on the report, the display should not be too crowded and not too spread out. Sufficient margins and spacing throughout the output will enhance readability. Desired information must be easy to locate. Comparisons, ratios, percentages, exception flags and graphs should be used where necessary.

D. Timeliness

Reports must be prepared and ready for use in time. Most reports provide information, which is used to make decisions. Hence, this information must reach the recipients while the information is pertinent to transactions or decisions. Information is of very little use if it arrives after the decisions are made.

E. Cost

Every report has two costs. First is the cost of preparation, which consists of analysis, design, computation and distribution. Second is !he cost of reading the report and locating germane parts of it. Often the cost of reading the report is forgotten during the calculation of costs. The reading cost can be significantly reduced only if the appropriate information is presented clearly on the report. The total cost should always be less than the expected benefits. Only then the report should be prepared.

SELF ASSESSMENT EXERCISE

1. What do you understand by external information?
2. Which criteria must be satisfied for a form to be attractive?

4.0 CONCLUSION

Types of information, criteria for form and report design were all considered in this unit.

5.0 SUMMARY

What you have learned in this unit involves criteria for forms and report design.

6.0 TUTOR-MARKED ASSIGNMENT

What are the three sub classes of internal information?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

MODULE 3 SYSTEM DESIGN, IMPLEMENTATION AND MAINTENANCE

UNIT 1 PHYSICAL FILE DESIGN AND DATABASE DESIGN I

CONTENTS

1.0 INTRODUCTION

In this unit we will introduce the concept of database design. We will equally consider the design of database fields as well as the rules for naming tables and fields.

2.0 OBJECTIVES

After going through this unit, you should be able to:

state the advantages of database over files
distinguish between logical and physical design
state the rules for naming tables and fields.

3.0 MAIN CONTENT

3.1 Introduction to Database Design

Over a period of time, massive advancements have taken place in the field of databases. Storage of data and retrieval is an integral part of any information system.

A. Flat Files VS. Database

Traditionally, data are being kept in flat files. Consider an example of flat file which stores pin code of customer's address. Any change in the size of the field will enforce changes in the entire program which uses the data related to customers' address. If the file is being used by more than one application, the problem can become even worse.

B. Steps in Database Design

Analysis is the process of creating a conceptual data model independent of the target database technology. The typical result is an ER model.

Design is the process of creating a logical data model. This step is dependent on the target technology (relational, hierarchical, or network), but not on the specific database implementation (such as Oracle, MS SQL, DB2 for OS/390 or DB2 Universal Database).

Implementation is the process of creating a physical model or schema for one specific database system, such as Oracle, MS SQL, and DB2. The result is an optimized physical design.

C. E-R Model to Database Design

The process of database design process involves the task of converting logical model to working physical model. The logical model can be arrived at by the application of nonJ1al forms. Normal forms are nothing but a set of rules applied sequentially starting from the basic table to the table resultant due to the application of a normal form. As of today, there are a total of five normal forms. The first normal form is applied to the basic table. The resultant table is given as input to the second normal form and so on. It is not mandatory to continue this process until the fifth normal form. Usually, the process is continued until third normal form. Fourth and fifth normal forms are applied only when there are multivalue dependencies and join dependencies respectively. We are not introducing you to the normalization theory in this course. It will be dealt in the course related to databases in detail. Physical database design starts from a given relational model. That is, from the definition

of a set of tables and their respective columns. The objective of physical database design is to fulfill the performance requirements of a set of applications by optimizing the use of the DBMS. Key areas include: optimizing the index configuration, data placement and storage allocation.

Entities with one-to-one relationship should be merged into a single entity.

A table models each entity with a primary key and non-key attributes, some of whom may be foreign key(s).

One-to-many relationships are modeled by a foreign key attribute in the table representing the entity on the "many" side of the relationship.

Many-to-one relationship between two entities is modeled by a third table that has foreign keys that refer to the entities. These foreign keys should be included in the primary key of the relationship table, if appropriate.

Many commercially available tools can automate the process of converting a E-R model to a database schema. Figure 9.1 depicts various steps involved in the design of databases.

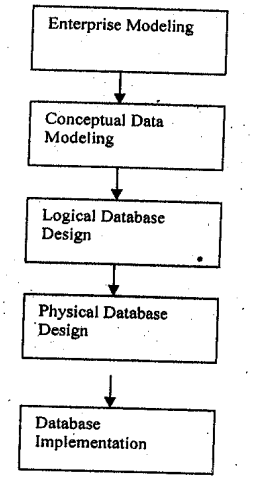


Figure 9.1: Steps in Database Design

D. Inputs to Physical Database Design

1. Logical structure of Database (Normalized relations).
2. Definition of attributes -data type, integrity control, error handling.

3. Choice of RDBMS:

a. Hierarchical
b. Network
c. Relational (DB2, MySQL)
d. Object relational (Oracle 8i/9i)

4. Estimation of database size growth rate and frequency of usage.
5. Requirements for backup, recovery, response time and retention time.

E. Guidelines for Database Design

The following are various guidelines for Database Design:

ensure that the data stored in the files (database tables) are atomic. Data stored in the atomic form can be combined later to generate data in specific form;
every table must have a primary key which identifies each record in the table distinctly. Descriptive and meaningful name is to be used while naming a field in the table (For example, use *product_id* instead of *ID);*
use single column primary key whenever possible. As *most* of the join operations are made: on primary key and composite primary keys make the operation slower;
use numeric key whenever possible;
use primary key name as foreign key for better readability;
avoid allowing null values to go into the columns that have discrete range of possible values; and
avoid multiple tables with similar structure when one table is sufficient.

Figure 9.2 depicts various records and fields in a file which consist of details of various employees in a hospital.

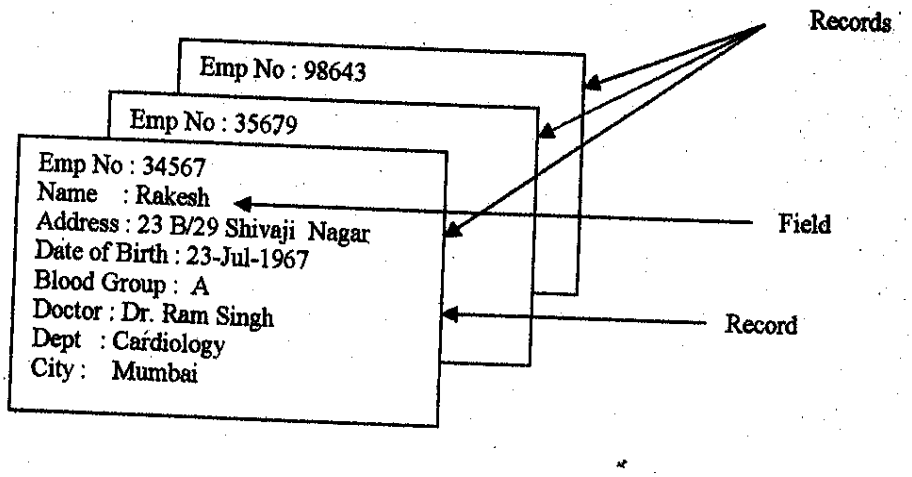


Figure 9.2: Records and Fields

3.2 Design of Database Fields

Attributes in E-R model are known as fields in physical data model. A field is the smallest unit of data that is stored and manipulated. Fields are used in conventional files as well as in databases. It is the implementation of attributes and can be termed as smallest unit of meaningful data.

3.2.1 Types of Fields

The following are various types of fields in databases:

Primary Key: Any conventional file system or database stores two kinds of field's namely descriptive fields and primary key. Descriptive fields comprise the customer names, inventory numbers, item descriptions, and so on, which are used by the application. Keys refer to the primary and foreign keys that are used to find database records and relate them to one another.

Keys are fundamental to the concept of relational databases because they enable tables in the database to be related with each other.

A table must have a primary key i.e. an attribute or combination of attributes that are guaranteed to be unique and not null It is sometimes helpful to introduce a surrogate field to act as a key. This could be a table attribute, which has no business meaning, but simply added to serve as a unique identifier for each record in the table. This is sometimes referred to as *plumbing.*

The requirements for a primary key are very hard. It must conform to the following rules:

They should exist.
Be unique in the table.
The values must not change or become null during the life of each entity instance
It must have a not-null value for each instance of the entity.

Surrogate keys are often required because sometimes, real business data does not fulfill the requirement of a primary key. Furthermore, the surrogate key is typically a single field (not a composite key), which simplifies the database schema, particularly when the key is used in other tables as a foreign key.

Most of modern RDBMS are tuned in for queries on integers, so it is advisable to use this datatype as a primary key. Many RDBMS provide a special serial number or sequence number of integer type, which generate a sequence of unique integers as a row is inserted into the table. Declaring a column to be of this type guarantees that a unique key is generated for each inserted row.

Secondary key: Also known as *Alternate key.* This is a field or collection of fields in the table which can be used as primary key in addition to the already existing primary key.

Foreign keys are table attributes, the values of which are the same as those of primary keys of another table. It is often desirable to label foreign key columns explicitly. For instance, by adopting a naming convention. Existence of foreign key enforces the referential integrity constrains (discussed later in this Unit). A referential integrity constraint (references) should be declared as part of the CREATE statement in a DBMS while creating the table.

Descriptive fields: Attributes that are not used as key but store business data.

3.2.2 Rules for Naming Tables and Fields

Names for all database elements should be:

Unique
Meaningful
Short

Restrictions for naming tables:

Use no acronyms or abbreviations. Should be descriptive to convey meaning.
Should not imply more than one subject

Restriction for naming fields:

No acronyms
Use abbreviations only if clear and meaningful
Should not imply more than one subject
Should be singular.

While designing database fields, it is required to set the properties of the fields.

Name: A name is used to refer the attribute in the DBMS that uniquely labels the field. The name of the attribute in the logical data model and the name of the field in he physical data model must be same. For example, *student name* in a *student* table.

Data Type: Type of data the field is expected to store. This could be numeric, alphanumeric etc. The data type, supported by various RDBMS varies to a great extent. For example, student_name CHAR(25), indicates that the name of the student is of character data type, 25 indicates the maximum size of the data that can be stored n the field. The data type selected should ensure the following:

it involves minimum usage of memory and represents all possible values
supports all types of data manipulation that is expected from the business transaction.

Size: It indicates the size of the database fields. Many RDBMS support sizes that are variable. For example, VARCHAR data type in Oracle.

Null or not Null: specifies whether the field will accept null value. Not null constrains applied in DBMS ensure that null values are not entered to the respective fields. A null value is a special value distinct from 0 or blank. A null value indicates that the value is either missing or unassigned yet. We may specify that customer_name in a customer table to be not null. When a field is declared a primary key DBMS automatically ensures that the field in not null.

Domain: It indicates the range of values that are accepted by the fields. For example: Basic_Pay in a *employee* table can assume any value between the lowest basic-pay and highest basic -pay existing in the company. In such cases, the .value of the field can be restricted to the

one between the highest and lowest value to avoid entry of non-existing basic_pay.

Default Value: It refers to the value that is stored by default in the field. For example, ship_date in a invoice is most of the time same as invoice_date (current date). When a default value is assigned to a field, it reduces a lot of data entry time and reduces the chances of error.

Referential Integrity: It refers to a set of rules that avoid data inconsistency and quality problems. Referential integrity ensures that a foreign key value cannot be entered unless it matches a primary key value in another table. RDBMS automatically enforces the referential integrity once the database designer identifies and implements primary and foreign key relationship.

It prevents orphaned records. i.e. when a row containing a foreign key is created, the referential integrity constrains enforced vide the RDBMS ensure that the same value also exists as a primary key in the related table.

When a row is deleted, it should be ensured that no foreign key in related tables is the same value as primary key of the deleted row.

Figure 9.3 depicts primary and foreign keys for two tables namely *customer* and *order*.

Customer_id	Customer_name	Customer_city	Customer_phone
5466	John	New Delhi	2345678
5678	David	Mumbai	2567890

Table Customer

Order_no	Order_date	*Customer_id*	Amount
123456	12/3/2004	*5466*	Rs 345
345678	11/3/2003	*5678*	Rs. 567

Table Order

Figure 9.3: Primary key and foreign key relationship

Customer _id is the primary key in customer table and is a foreign key in order table. This referential integrity constraint ensure that the value *customer_id* in order table must exist in the customer table. The primary key is shown in bold-and the foreign key in italics.

SELF ASSESSMENT EXERCISE

1. Describe a primary key.
2. A secondary key is also referred to as…….

4.0 CONCLUSION

Database design, design of database fields, types of fields and rules for naming tables and fields were all considered in this unit.

5.0 SUMMARY

What you have learned from this unit borders on the physical file and database design.

6.0 TUTOR-MARKED ASSIGNMENT

Define referential integrity.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com
http;//www.cbpa.edu/flin/info609/sysplan

UNIT 2 PHYSICAL FILE DESIGN AND DATABASE DESIGN II

CONTENTS

1.0 INTRODUCTION

In this unit, you will learn about the design of physical records, types of files and design of database. We will equally consider a case study.

2.0 OBJECTIVES

After going through this unit, you should be able to:

describe how to design field record in a database table
understand various constrains enforced during database design
explain the concepts of records, record types and files as well as different techniques for placing records on disk.

3.0 MAIN CONTENT

3.1 Design of Physical Records

A Record is a collection of fields. Records are common to both databases and files. Records are collection of fields in a predefined format. The design of physical record involves putting the collection of fields in a single logical unit so that the fields are stored in adjacent locations for better storage and retrieval.

The main objective of the design of physical records is to store and retrieve them efficiently. Also, the fields should be stored in adjacent locations in such a way that the storage is used efficiently and speed of data processing is appropriate.

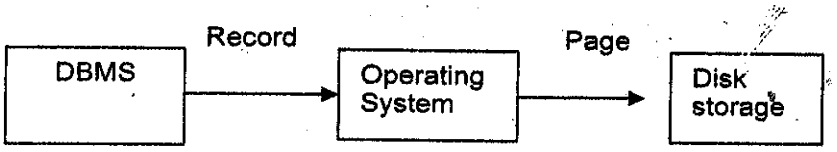


Figure 9.4: Process of storing a record physically

Physical pages or blocks are units of information moved between disk and memory buffers. They hold not only records, or table entries, but other information such as the amount of free space currently available in the block, the starting position' of each record, etc. Blocks of data (pages) are normally read or written by the operating system. Page is referred to as the amount of data written in one I/O operation of operating system.

Blocking factor refers to the number of physical records per page. If a record size is 1340 bytes and the page size is 2048 bytes, then 708 bytes are wasted if DBMS does not allow physical records to span different pages. Selecting a block size involves a trade-off. In principle, the larger the block size, the fewer read-write operations need be performed to access a file by the operating system and therefore the more efficient is the processing. However, it requires a correspondingly large allocation of buffer space in memory. Since this is limited (and perhaps shared by many users), there is in practice, an upper bound. Moreover, large block sizes are primarily advantageous for sequential access.

Denormalization is the process of transforming normalized relations into unnormalized physical record specifications. The motivation behind de-normalization is poor performance of normalized table. The following may be of use for denormalization.

Combine two entities with one-to-one relationship to one entity. This avoids the cost of joining two tables when the data are required from both the tables.

Another form of de-normalization is to repeat the non key attribute (field) of one table in another table to facilitate the execution of query faster. However, it depends on the application at hand. Figure 9.5 depicts denormalization for optimized query processing.

Order_no	Order_date	*Customer_id*	Amount	Customer_name
123456	12/3/2004	*5466*	Rs 345	John
345678	11/3/2003	*8909*	Rs. 567	David

Figure 9.5: Denormalization for Optimized Query Processing

In a particular application it is seen that queries about order also requires the cutomer_name. In case of nomlaiized table, this would always require joining Customer table and order table each time the query is processed. We have modified the order table by adding back the customer_name from the customer table in order table. Now all queries will require only the order tables as all relevant information are available in this table.

Activities to enhance performance

1. Combining tables to avoid joins
2. Horizontal partitioning refers to placing different rows of a table into separate files. For example, in an order table order pertaining to different regions can be kept in a separate table for efficient retrieval of records.
3. Vertical partitioning refers to placing different columns of a table into separate files by repeating the primary key in each of the files.
4. Record partitioning refers to a combination of both horizontal and vertical partitioning as in distributed database processing.
5. Data replication refers to the same data being stored in multiple places in the database.

Figures 9.6 and 9 7 depict table's structure in its original and improved versions.

Process of Denormalization of Tables

Select the dominant process based on frequency of execution and frequency of data access;
Define the join table for dominant process;
Evaluate the cost of query, updates and storage for the schema. Consider possibility of renomlaiization to avoid table joins

Fixed Length Records and Variable Length Records

In fixed length record format, the length of record is always the same. The fixed length records are easier to design and implement but are more wasteful than variable length record formats when comes to utilization of space.

In variable length record format, the records are of variable length. To identify the end of a record, variable length records use end of record marker. It can be a special character.

Name	Date of Birth	Address	Phone
Rakesh	23-JUL-1967	12/B Patel Nagar, New Delhi - 110023	2234567
Ahbinav	8-NOV-1970	1201 Erose Appt., New Delhi - 110001	2236780
Roy	1-APR-1954	M-1034 Vanasthalipuram, Hyderabad-500001	2438723
Venkat	5-OCT-1969	24/A Gandhinagar, Vijayawada-520003	2658913
Ajay	5-AUG-1975	56 Canal Road, Patna- 600005	2753219
Sachin	8-SEP-1969	234 Mount Road, Varanasi-546987	2658703

Figure 9.6: Structure of Students table

Name	Date of Birth	Address	City	Pin Code	Phone
Rakesh	23-JUL-1967	12/B Patel Nagar	New Delhi	110023	2234567
Ahbinav	8-NOV-1001	1201 Erose Appt	New Delhi	110001	2236780
Roy	1-APR-1954	M-1034 Vanasthalipuram	Hyderabad	500001	2438723
Venkat	5-OCT-1969	24/A Gandhinagar	Vijayawada	520003	2658913
Ajay	5-AUG-1975	56 Canal Road	Patna	600005	2753219
Sachin	8-SEP-1969	234 Mount Road	Varanasi	546987	2658703

Figure 9.7: Improved Table Structure of Figure 9.6

3.2 Design of Physical Files

Files are collection of logically connected records. In RDBMS, files are called tables. However, the way the files are stored in memory depend on the operating system. Many operating systems allow files to be split into pieces but the same is transparent to the user.

3.2.1 Types of Files

A Master file is a permanent file of all the data needed by a business. Some of the fields may be regu1arly updated from transactions. For example, a file consisting of information about customers.

A Transaction File is a temporary .file of all the transactions (items bought, sold, deleted etc.) that have taken place ill a given period. It stores data related to day to day business transactions. For example, data related to daily sales activity. The contents of these files are used to update the master file. Daily sales are used to update balance in the customer file.

An Archive File is a file of data in permanent storage (usually, the data is stored for legal reasons or to perform trend analysis). Archive files will contain all information about the past dealings of the business and would normally be stored on a different site to facilitate recovery in case of a disaster such as fire.

An Audit File is a file that does not store business data but data related to transaction log. For example, data and time of access, modification etc. of data, values of fields before and after modification etc.

A Work File is file temporarily created to hold intermediate result of the data processing For example, a sorted file of list of customers.

3.2.2 File Organization

File organization is the physical organization of records on the disk. There are different types of file organizations depending on the organization of records in the disk and other secondary storage.

Before deciding on a specific file organization, we should ensure that its application leads to the following:

Fast retrieval of records
Reduce disk access time
Efficient use of disk spaces.

Serial File Organization

A serial file is created by placing the record as it is created. It leaves no gap between the records that are stored on the disk. The utilization of space called packing density approaches 100 percent in this case. Examples of serial files are print file, dump file, log files, and transaction files. These files are created once and are not used for addition or deletion or any kind of record searching operation.

Sequential File Organization

In this organization, the records are physically ordered by primary key. To locate a particular record, the program starts searching from the beginning of the file till the matching primary key is found. Alphabetic list of customers is a common example of sequential file organization. Deletion of record may cause wastage of space and adding a new record requires rewriting of the file. This type of file organization is suitable for master files and is not used where fast response time is required.

Indexed Sequential File Organization

In this organization, records are not physically ordered. Index is created to facilitate searching of regards. Index Records give physical location of each data record. Indexes are separate files with a link to the main file. This type of file organization is used where faster response time is required. Figure 9.8 depicts Indexed sequential file organization.

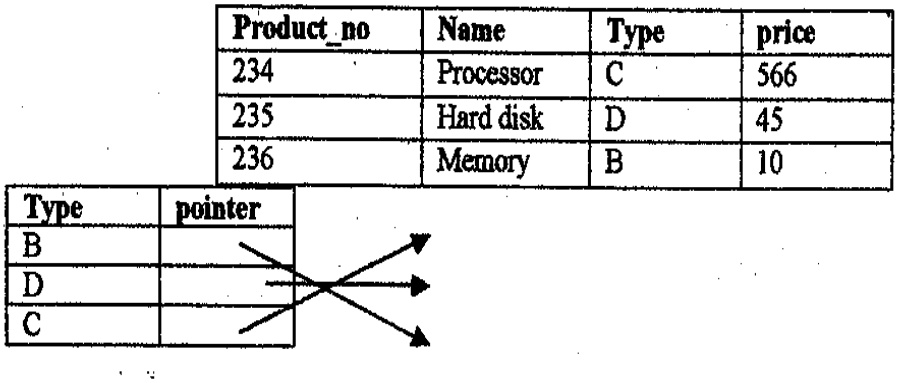


Figure 9.8: Indexed Sequential File Organization

Hashed File Organization

In this organization, records are physically ordered according to hashing algorithm. The address where each record is stored is determined using hashing algorithms. Figure 9.9 depicts an example of a Hashed file organization.

The following is a typical hashing algorithm:

1. Uses a field in record called the hash field (generally the key filed).
2. Divides by prime number known as hash function.
3. Produces record address called the hash address.

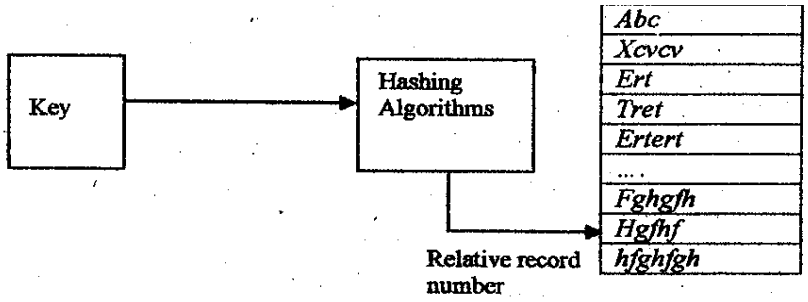


Figure 9.9: Hashed File Organization

3.3 Design of Database

Database design is similar to the pillars of a building. Any negligence, errors in Database design may lead to degraded performance of the software. In some cases such as real time applications, it may also lead to disasters.

The following are various steps in Database design:

Selection of database architecture
Designing database schema
Selecting indexes
Estimating capacity of the database.

Selection of Database Architecture

Selecting database architecture is one of the most challenging parts of database design for any information system. Before deciding on the target DBMS where the database is to be implemented, few considerations are required.

Hierarchical database structure is a kind of database management system that links records in tree data structure such that each record has only one owner. For example an order is owned by only one customer. It resembles a tree structure.

Network database structure is more flexible structure than Hierarchical model as well as relational model, but not preferred due to the high processing time. A neural network is an example of network database.

Relational database structure is most commonly used database model. The data is modeled as a mathematical relation. Reference key joins different tables together. Indexes provide rapid access to specific record in the database. For example, DB2, MySQL, Oracle are some RDBMS.

Object Oriented Database management system is based on object oriented paradigm. In object oriented database, data is stored as objects and can be interpreted only by using the methods specified by its class.

A blue print of the database is a physical model. A schema defines the database in terms of tables, keys, indexes and integrity rules. A relational schema consists of a relation, name of the attributes in the relations and restrictions on the relations called integrity constraints.

A **database schema** is a set of relation schemas. Changes to a schema or database schema are expensive. So, careful thought must be given to design of a database schema. The following are some guidelines for the design of a database schema.

Each entity should be implemented as a database table.
Each attribute should be implemented as a field.
Each table must have a primary key and an index based on the key.
Each table may have zero or more secondary keys.
Appropriate foreign keys.

The following is an example of a database schema:

employee (emp_name, birth_date, social_security_no, dept_name), department(dept_name, function).

Figure 9.10 depicts the database schema of the above.

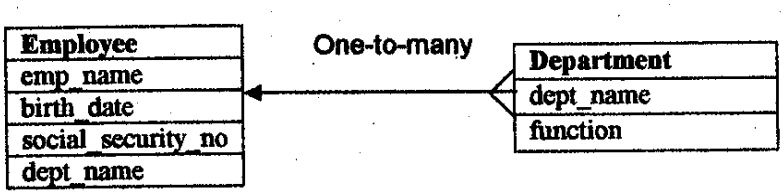


Figure 9.10: A Database Schema

A database schema defines a database in terms of tables, keys, indexes and constraints.

The following are various objects of a schema:

Fields
Records
Tables
Database

Selecting Indexes: During the process of file and database design one must choose index based on a single field (usually the primary key) or multiple fields. While selecting index, one must keep in mind the performance issues vis-a-vis the issue of inserting and deleting records. While indexes can be used generously on tables primarily used for query purpose with rare necessity to update records, they should be used judiciously in tables that support transaction processing which involve large insertion, updation and deletion operations. The amount of time needed to maintain the indexes in database tables increases with the number of rows stored.

When an index is created on a table, a separate storage area is allocated to store the index structure. A database table can have one or more indexes associated with it.

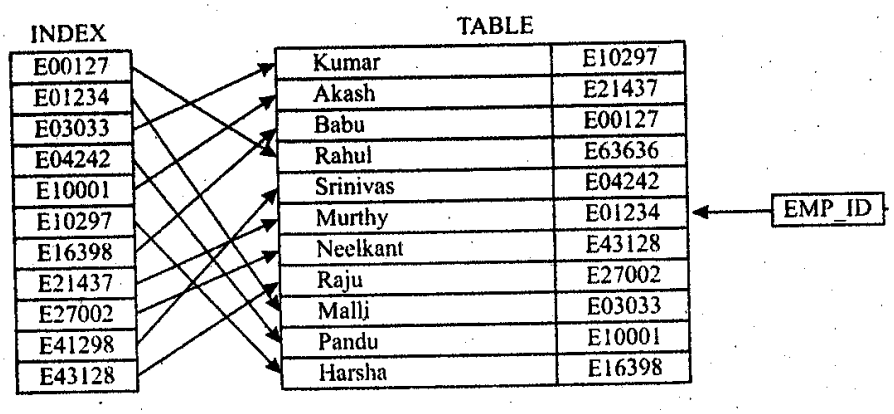


Figure 9.11: An index created on the EMP _ ID Field

For example, consider an employee record. Figure 9.11 depicts an index created on the EMP_ID field of the employee table. The index table contains a sorted list of the employee ID values. Indexes may significantly improve the performance of SQL queries. It may not be noticed with small tables but it can be quite significant for larger tables. However, there are disadvantages to having too many indexes on table. Indexes can slow down the speed of some inserts, updates, and deletes when the

DBMS has to maintain the indexes as well as the database tables. Also, indexes take additional disk space as they are stored separately.

Example

In an employee database table, one will retrieve records based on employee name, department, or hire date. One may create three indexes-one on the DEPT field, one on the LAST NAME field, and one on the HIRE DATE field. The indexes are created on fields that appear in **where** clause of select statements.

If the Boolean operator in the where conditions is AND (for example, CITY = ‘Mumbai’ AND STATE = ‘MH’), then a concatenated index on the CITY and STATE fields will help. This index is also useful for retrieving records based on the CITY field.

If the Boolean operator in the where condition is OR (for example, DEPT = ‘EDP’ OR HIRE_DATE> ‘0/30/03’), an index does not help performance. Therefore, index file need not be created.

Estimating capacity of the database: Database administrator needs to calculate the amount of disk space required for the database.

1. In a given table, calculate the record size by adding the sizes of various fields in it.
2. Also, the number of records that will be present in each table at a particular period of time should be forecast.

Table size = record size * number of records in the table.

Database size is sum of sizes of all tables in that database. As rule of thumb, add a factor of 50% for indexes and other overheads to get the expected database size.

While designing a database, future growth of database should also be kept in mind. Most of the business databases have a liner growth trend. Figure 9.12 depicts the growth of a database in relation to time.

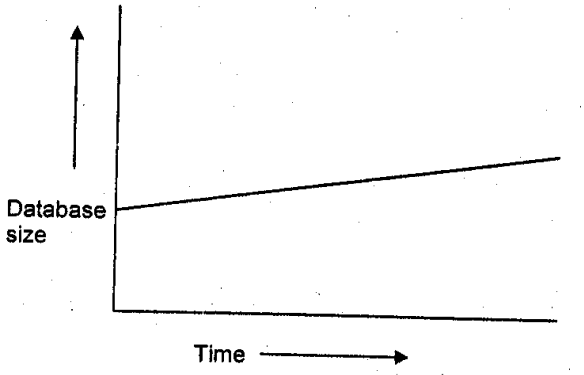


Figure 9.12: Growth of Database

The physical database design is the process of transforming a logical data model into an actual working physical database. A logical data model is required before designing a physical database. We will assume that the logical data model is complete, though. We will consider physical database design targeting a relational DBMS.

The very first step is to create an initial physical data model by transforming the logical data model into a physical implementation based on the target DBMS to be used for deployment. In reality, the physical data model depends largely on the target DBMS to be used for implementing the database. Therefore, to successfully create a physical database design requires a good working knowledge of the features of the DBMS including:

knowledge of the database objects supported by DBMS, the physical structures' and files required to support those objects;
details regarding how the target DBMS supports indexing, referential integrity, constraints, data types, and other features that augment the functionality of database objects;
knowledge of the DBMS configuration parameters and limitations; and
data definition language (DDL) to translate the physical design into actual database objects.

We can create an effective and efficient database from a logical data model, i.e. E-R Diagram. The first step in transforming a logical data model into a physical model is to perform a simple translation from logical terms to physical objects. It maybe noted that this simple transformation will not result in a complete and correct physical database design it is simply the first step and can act as a template to further refining the database design. The transformation consists of the following steps:

transforming entities in ER diagram into database tables;
transforming attributes into table columns; and
transforming domains into data types and constraints to primary key and foreign key relationship.

Deciding a primary key is an integral part of the physical design of entities and attributes. A primary key should be assigned for every entity in the logical data model. One should try to use the primary key as selected in the logical data model. However, if suitable attribute is not available, multiple attributes can be used as primary key. It may be required to choose a primary key other than the attributes in logical design by inserting a column that does not store business data (surrogate key) for physical implementation.

In a physical database, each table column must be assigned a data type supported by the DBMS. Certain data types require a maximum length to be specified, e.g., a character data type could be specified as CHAR (25), indicating that up to 25 characters can be stored in the column.

Figure 9.13 depicts the E-R diagram of an employee database.

After following normal steps of transformation described above, further refinement of relations is possible before implanting on the target DBMS. One such example is shown below.

In the above ER diagram, Department being a multivalued attribute, we will convert this multivalued Attribute into relation namely Department.

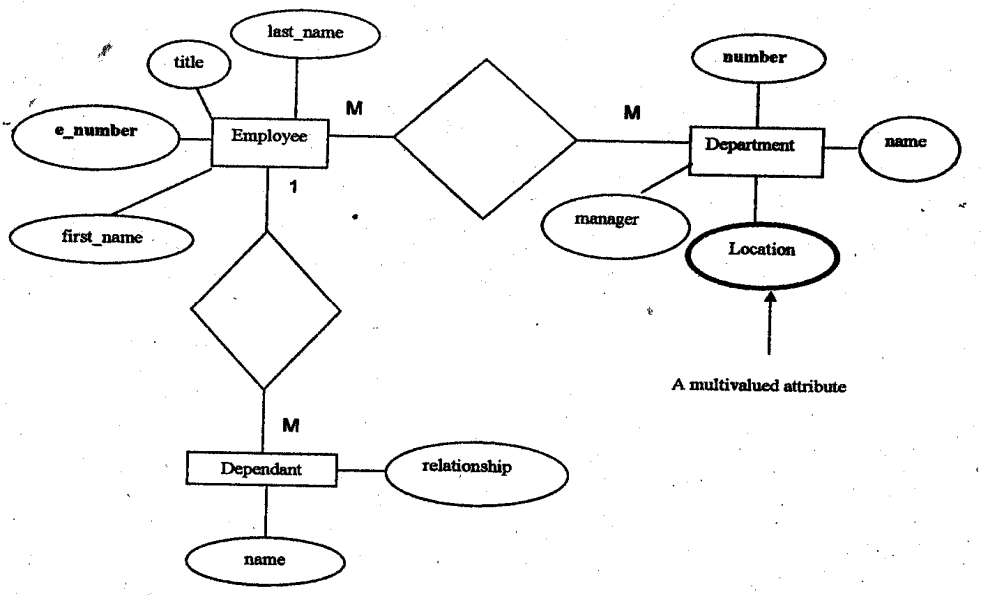


Figure 9.13: E-R Diagram for Employer database

e_number	Title	Last_name	First_name	Supervisor_emp_no

EMPLOYEE

number	Name	Manager

DEPARTMENT

Dept_no	Location

LOCATION

Emp_number	Name	relationship

DEPENDENT

Figure 9.14: A Relational Database Schema

Figure 9.14 shows the nearest relational database schema for the E-R diagram depicted in Figure 9.13. The aim of database design should be to create a database plan to fit present and future objectives. A logical data model should be used as the blueprint for designing and creating a physical database. But, the physical database cannot be created properly with, a simple logical to physical mapping. Physical database design decisions need to be made by the Data Base Administrator before implementing physical database structures to the target DBMS. Sometimes, this may require deviation from the logical data model.

SELF ASSESSMENT EXERCISE

1. List any three types of files.
2. What do you understand by file organization?

4.0 CONCLUSION

In this unit, you have learned how to design field and records in a database table. You have also considered different techniques for placing records on it.

5.0 SUMMARY

We have learned more aspects about the physical file design and database design in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

Define a master file.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com
http;//www.cbpa.edu/flin/info609/sysplan

UNIT 3 CASE TOOLS FOR SYSTEM DEVELOPMENMT I

CONTENTS

1.0 INTRODUCTION

CASE involves using software packages called CASE tools to perform and automate many activities of system development life cycle.

The use, role and types of CASE tools are considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

know the role of CASE tools and their use by organizations
know various components of CASE tools
list the classification of CASE tools.

3.0 MAIN CONTENT

3.1 Definition of CASE Tools

A CASE tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process.

Although, ideally a CASE tool should support all the activities of software engineering process starts from requirements analysis to designing, coding, testing, implementation and documentation, in reality, CASE tools often support one activity or at least a group of related activities.

As software development activities become more complex and relatively unmanageable, there has been an awareness of the need for automated tools to help the software developer to accomplish this task. Initially, the focus was primarily on program support tools such as design of translators, compilers, assemblers, macro processors, and other tools. As computers became more powerful and the software that ran on them has grown larger and more complex, the support tools began to expand

further. Some capabilities of CASE tools are also found in the common application development software.

Large-scale use of computers has necessitated its maximum and efficient use and development of software for various activities of any organization. A software development effort can be viewed as a significant effort to design appropriate solutions, test and implement the solutions and finally documenting the solutions. In view of this, a wide range of support tools began to emerge to help the development team.

3.2 Use of CASE tools by organizations

The following are some of the ways in which CASE tools are used:

To Facilitate Single Design Methodology: CASE tools help organization to standardize the development process. It also facilitates coordinated development. Integration becomes easy as common methodology is adopted.

Rapid Application Development: Organizations use CASE tools to improve the speed and quality of system development.

Testing: CASE tools help to ease and improve testing process through automated checking and simplify program maintenance.

Documentation: In traditional software development process, the quality pf documentation at various stages depends on the individual. CASE tools improve the quality and uniformity of documentation at various stages of SDLC. It also ensures the completeness of the documentation.

Project Management: It improves project management activity and to some extent automates various activities involved in project management.

Productivity and Reduction of Cost: Use of CASE tools makes the software easy to maintain and hence reduce the maintenance costs. Automation of various activities of system development and management processes increases productivity of the development team.

3.3 Role of CASE Tools

CASE tools play a major role in the following activities:

Project management

Data dictionary
Code generation
User interface design
Schema generation
Creation of meta-data for data warehouse
Reverse engineering
Re-engineering
Document generation
Version control
00 analysis and design
Software testing
Data modeling
Project scheduling
Cost estimation

CASE technology has resulted in significant improvements in quality and productivity. An ideal CASE tool should support all facets of system development like analysis, design, implementation, testing and maintenance. All aspects of software engineering process are not supported by today's CASE tool. Most of the CASE tools provide good support for data modeling, object oriented design and programming. Also, they moderately support testing and maintenance.

3.4 Components of CASE

The activity that can be automated, whether partially or fully, depends on the CASE tools that are used. Most of the CASE tools generate a working model or prototype, which makes development process faster and easier.

3.5 Types of CASE Tools

The following are various types of CASE tools:

Planning and Management Tools: Begin the development process with information planning and project management.

Analysis Tools: These tools ensure that business requirements are correctly captured during the analysis phase early in the development process. Analysis tools are used to check for incomplete, inconsistent or incorrect specifications.

Design Toolset: It provides detailed specification of the system.

Information Integrator: It integrates system specifications and checks them for- consistency and completeness. It also records them in the CASE repository.

Code Generator: It automatically generates code specific to a language based on the system specification.

Database Design Toolset: It suggests database design and generates system control information.

User Interface Generator: It generates user interface based on system specification.

Report Generator: It generates reports based on specification

Figure 10.1 depicts various components of a typical CASE tool.

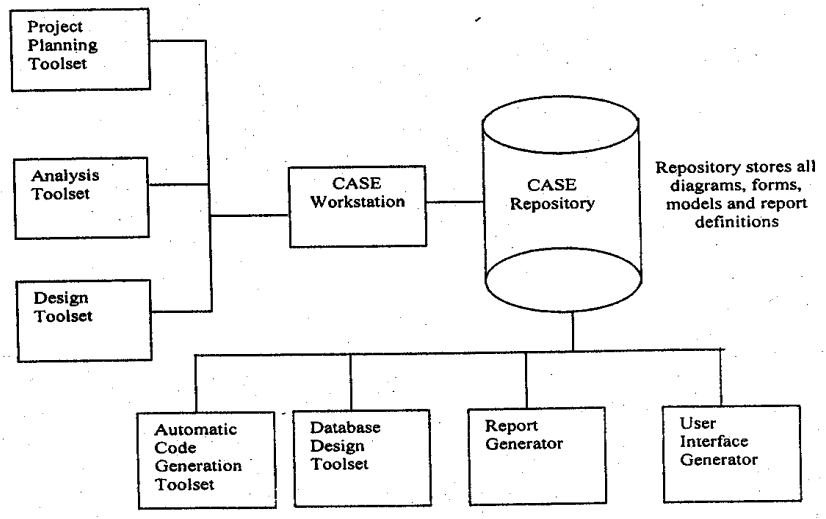


Figure 10.1: Components of a typical CASE tool

All case tools are based on prototyping which is particularly useful when the user requirements are difficult to define. Large systems use traditional SDLC approach but part of the system can be prototyped. The prototype is then repeatedly refined till it becomes acceptable.

3.6 Classification of CASE Tools

Figure 10.2 depicts various types of CASE tools.

Although CASE tools can be classified depending on the functionalities, these can be broadly classified into five generic categories:

Development Tools: These tools are interactive in nature. They are used for design support and code generation.

Front-End Tools: They support activities early in the life cycle of a software development process (planning, analysis and design). Examples are data flow diagram, data structure diagram, ER diagram, prototyping tools, etc.

Back-End Tools: They support activities later in the life cycle of a software development process (Implementation and maintenance). Examples are program flow chart, program editor, debugger, code generators etc.

Horizontal Tools: These tools are not specific to a particular life cycle step but are common across a number of life cycle steps e.g., Documentation tool.

Vertical Tools: These tools are specific to a life cycle.

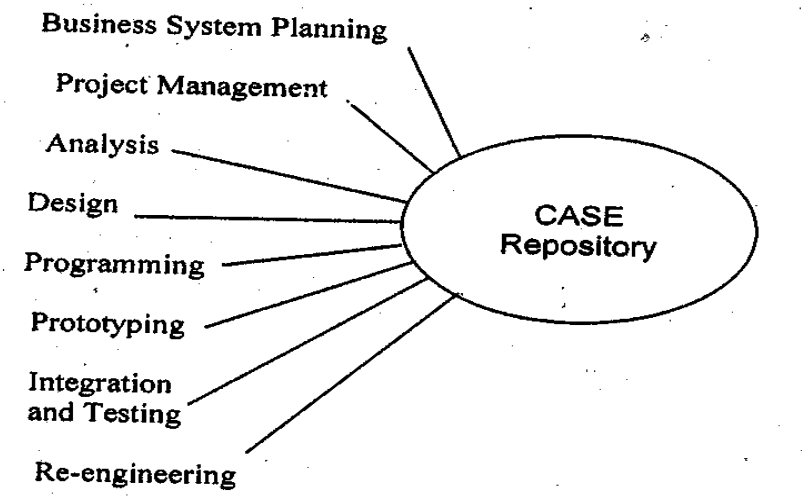


Figure 10.2: Types of CASE Tools

3.7 Reverse and Forward Engineering

We will discuss two important concept related to CASE tools namely, Forward Engineering and Reverse Engineering. Figure 10.3 depicts both Forward and Reverse Engineering.

Reverse Engineering is the process of recreation of model based on existing code. First, the existing code is scanned to generate the model. Then the model can be fine tuned in accordance with requirements. Reverse engineering allows developers to create model for old systems, which were never modeled. It analyses existing software with purpose of understanding its design and specification. Reverse engineering tools

read program source code and create graphical and textual representation of design.

Forward Engineering is the process of generation of skeleton code out of the models. First step is to create the model for a system, then generate the relevant code for the model and then allow modification of this code in tune with the requirements. *Re-engineering* means "restructuring and rewriting the legacy system or part of it without changing its original functionality". Re-engineering efforts make the software up-to-date to current technology and hence easy to maintain. The new system becomes restructured and re-documented. Re-engineering tools read program source code and interactively change and existing system to improve quality performance or maintainability.

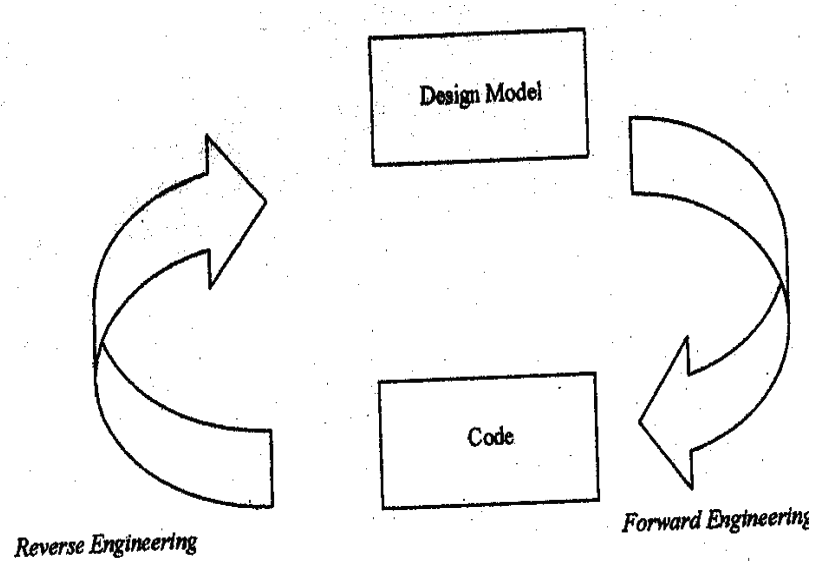


Figure 10.3: Reverse Engineering and Forward Engineering

4.0 CONCLUSION

In this unit, you would have learned about CASE Tools, their use, role and classification.

5.0 SUMMARY

What you have learned in this unit concerns CASE Tools.

SELF ASSESSMENT EXERCISE

1. Describe forward engineering.

2. What are five genetic categories of CASE Tools?

6.0 TUTOR-MARKED ASSIGNMENT

Define CASE Tools and give three uses of CASE Tools by organizations.

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 4 CASE TOOLS FOR SYSTEM DEVELOPMENMT II

CONTENTS

1.0 INTRODUCTION

Features of traditional systems development and CASE-based systems development are introduced. In this unit we will also consider emerging CASE tools as well as use of object oriented CASE tools to create reports and executable prototype.

2.0 OBJECTIVES

After going through this unit you should be able to:

state the features of traditional and CASE-based systems development
give typical examples of visual CASE tools
classify CASE tools, as front-end tools or back-end tools.

3.0 MAIN CONTENT

3.1 Visual and Emerging CASE Tools

Visual CASE tools enable user to quickly create user interface and related skeleton code.

3.2 Traditional Systems Development and CASE Based Systems Development

The following are the features of Traditional systems development:

Emphasis on program coding, testing and documentation
Specifications are to be written by the analyst and are paper-based
Coding is manual and tedious
Documentation is written by the programmer or the analyst and are often done after completion of the process
Software testing process follows the traditional approach
Manual maintenance of code and documentation.

The following are the features of CASE-based systems development:

Emphasis is on analysis and design
Rapid and interactive prototyping of models
Automated code generation
Automated documentation generation
Automated design checking
Maintain design specifications

3.3 CASE Environment

The earlier generation of CASE tool developers concentrated to a large extent on the automation of isolated tasks of system development process, such as document production, version control of source code, and design method support. Need of integrating these tools to support a common development environment was felt to support the activity effectively. A typical CASE environment consists of a number of CASE tools and related components for supporting most or all phases of system development life cycle that operates on a common hardware and software platform. CASE-environment is not just a random amalgamation of CASE tools, it provides proper interaction between the CASE tools. One should concentrate less on which components should be chosen, and much more on how the selected components can be made to work together effectively. Figure 10.4 depicts typical CASE environment.

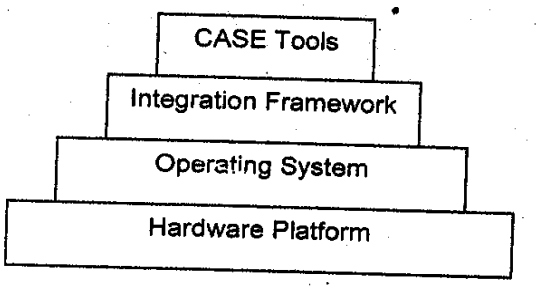


Figure 10:4: CASE Environment

The key to CASE tools is integration. Tools are more effective if they are integrated to work together. Integration could be data integration, user interface-integration or activity integration.

Examples of visual CASE Tools

Oracle 2000 Designer
Evergreen EasyCase

Aonix: Software Through Pictures
Popkin System Architect
Cadre Teamwork
ColdFusion
Rational Rose Visual
CASE Enterprise
Architect.

Rational Rose is one of the most widely used CASE tools by the software community. The teams responsible for software development are finding that modeling using CASE tools are becoming increasingly important for the software development process. Software developed using model driven technique such as Rational Rose offers a range of products to suit different activities of software development process.

UML Modeling: The Unified Modeling Language or UML is mostly a graphical modeling language that is used to express designs. It is a standardized language in which artifacts and components of a software system can be specified. It is important to understand that UML simply describes a notation and not a process. It does not put forth a single method or process of design, but rather is a standardized tool that can be used in a design process.

The following are some of the tasks that can be performed by using CASE tools:

UML modeling
Code generation/construction for Visual C++, Visual Basic, C++, Ada, JAVA Database design
Fully executable codes for C, C++ ,etc. across platforms
Component testing,

Another CASE tool is Enterprise Architect (Parx System). This is an object oriented CASE tool for the entire software development life cycle. Its features are:

Business process modeling
Forward and reverse engineering
Automation interface
Support for C++, Java, VB, VB.Net, Delphi
Project estimation tool

Testing tool
User interface design
Requirement gathering
Components model
Deployment model

3.4 Emerging CASE Tools

Initially CASE tools were not very sophisticated in terms of the process they support. Now, integrated CASE tools have emerged to support the entire gamut of system engineering and software development process.

Integrated CASE (I-CASE)

It offers automated systems development environment that provides numerous tools to create diagrams, forms and reports.
All tools share one common user interface.
User has the feeling of working on one tool.
Provide analysis, reporting and code generation facilities.
Seamlessly shares and integrate data across and between tools.
Repository is a central place to store information that is to be shared between various tools.

3.5 Object Oriented CASE Tools

Object oriented CASE tools are similar to other CASE tools. These differ from others only in terms of their capability to create class diagrams and text specifications for reports. Object oriented CASE tools support an 0-0 methodology such as *Rumbaugh's Object Management Technique (OMT).*

Examples of Object Oriented CASE Tools

A large number of object oriented CASE tools are available in the market. These include: *Paradigm Plus* from Protosoft, *Rational Rose* from Rational and *WithClass* from MicroGold Software. Our discussion here will describe object oriented CASE tools in general without making references to a specific product.

The following are some of the features found across most of the Object Oriented CASE tools:

Create graphics, such as class diagrams, message diagrams, state diagrams etc.
Create text specifications such as system specification, class specification and relationship specification.
Generate source code.
Repository of models.

Differences between Various Types of Object Oriented CASE Tools

All of these object oriented CASE tools are similar in terms of their capabilities to create class diagrams, code generation. However, they may differ in terms of their extendibility, number of supported operating systems and additional features and capabilities, such as support for different methodologies and computer language code generation (C++, Ada, Java etc). Most provide capability to automatically generate code of C++ and other languages from a class diagram and class specifications. Some provide the capability to generate class diagrams from code (reverse engineering). Figure 10.5 depicts drawing and text tools.

An object oriented CASE tool has the capabilities of drawing class diagram and state diagrams. Specification tools create text reports.

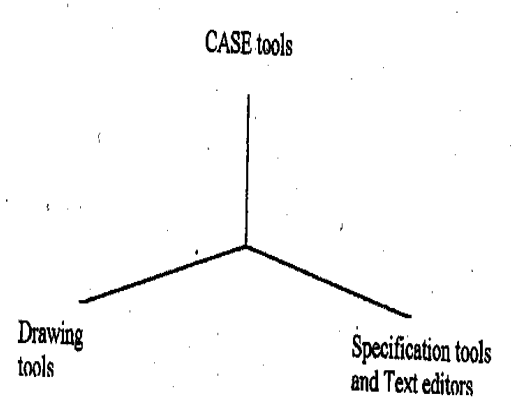


Figure 10.5: Drawing and Text Tools

Figure 10.6: depicts major outputs of object oriented CASE tools

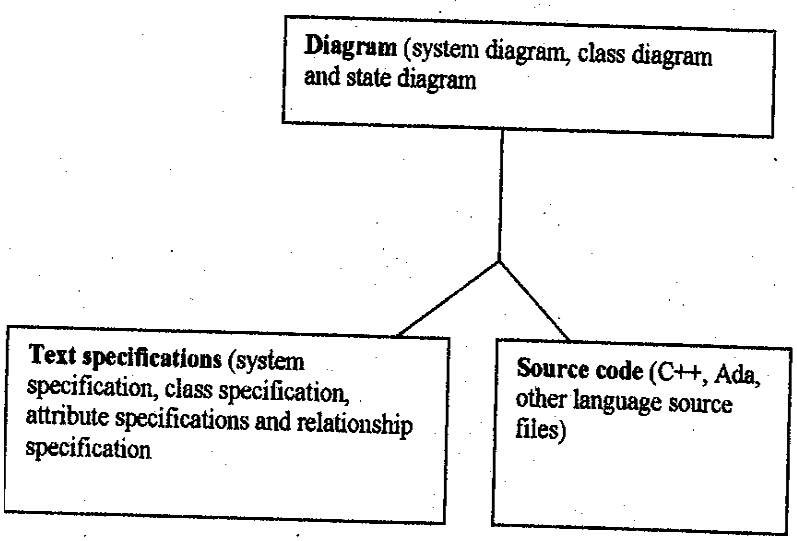


Figure 10.6: Major Outputs of Object Oriented CASE Tools

The class diagram is core to object-oriented design. It describes the types of objects in the system and the static relationships between them. The core element of the class diagram is the class. In an object-oriented system, classes are used to represent entities within the system. Entities are often related to real world objects.

Figure 10.7 depicts the class diagram for a typical Banking application.

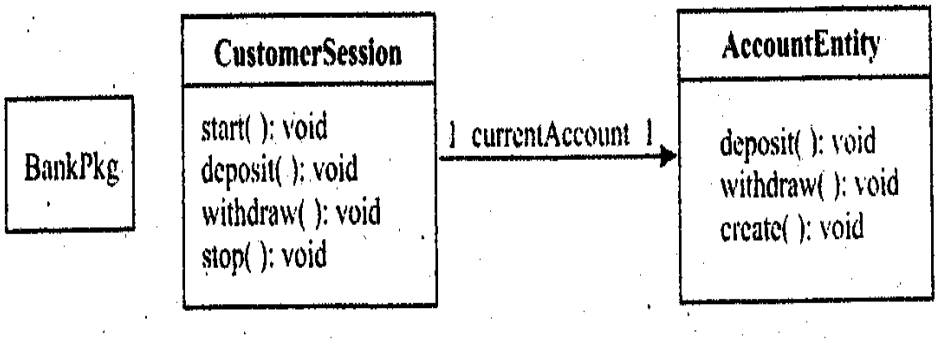


Figure 10.7: A Class Diagram for a typical Banking Application

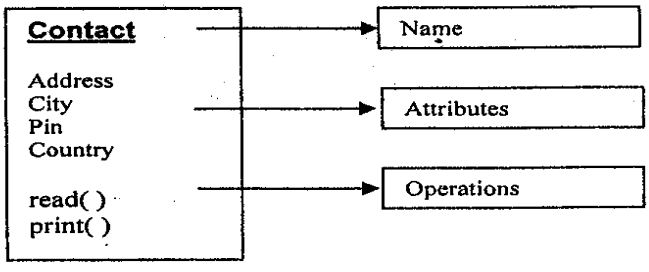


Figure 10.8: An Example of a Simple Class Contact

3.6 Creating Documentation and Reports

A system requirement describes a condition or capability which a system must conform to, either directly from the user need or derived from or stated in a contract, standard, specification, or other formally imposed document.

Most of the object oriented CASE tools assist in documenting as well as in object oriented analysis and design. Object oriented CASE tools have capability to import graphics from other tools. The system requirement is given shape in sketches with pencil and a paper. Then, it is used by the drawing tools and text editor available to create system diagram, class diagram and other diagrams. Document processor does it all to create a model document from these models. Figure 10.9 depicts the process to create a model document.

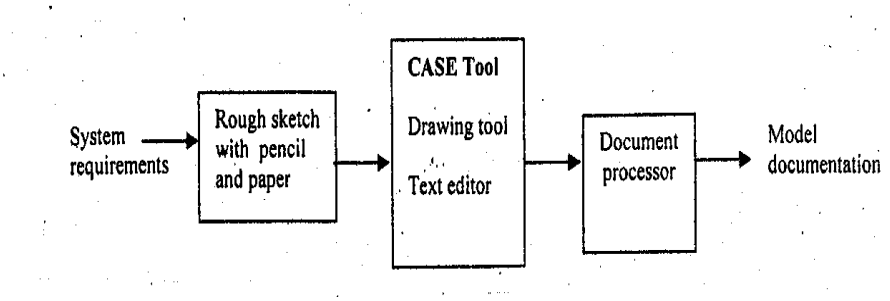


Figure 10.9: Process to create a model document

3.7 Creating an Executable Prototype

Most of the object oriented CASE tools greatly assist in creating executable prototypes based on design specifications. The diagram below shows how CASE tools are used to create executable codes. System requirements are prepared in text and diagrams by pencil and paper. Design specifications like class specifications, system diagrams and various text specifications are then generated using tools available in CASE. From this, design specification codes are generated using code generation tools. Most of the CASE tools support generation of C++ code whereas some may support other languages as well. The source code generated might require updation with formulas, expression, etc. CASE tools also create updated diagrams based on pda1ed source code. Figure 10.10 depicts the process to create an executable prototype using a CASE tool.

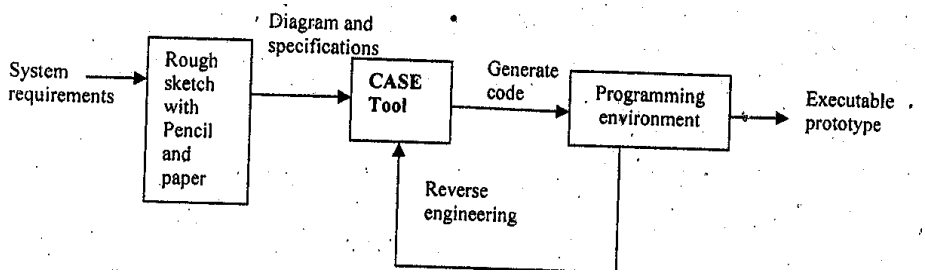


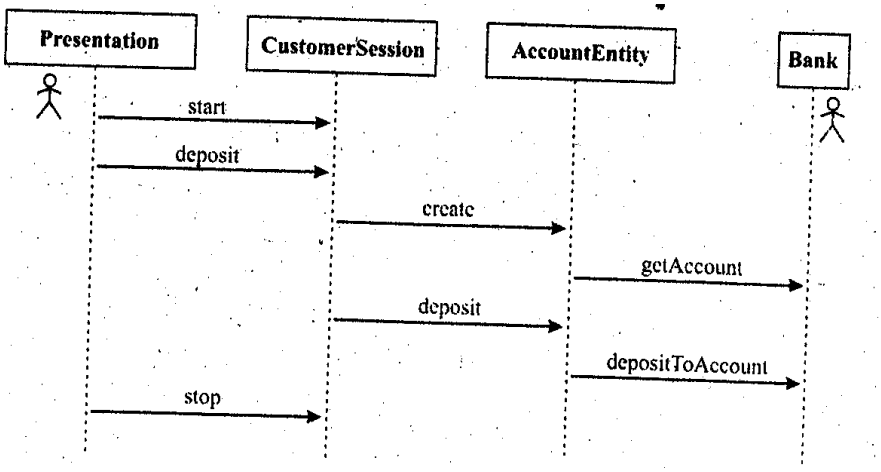
Figure 10.10: Process to create an executable prototype using a CASE tool.

3.8 Sequence Diagrams

UML provides a graphical means of depicting object interactions over time in Sequence Diagrams. These typically show a user or actor, the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case scenario or flow of events.

Sequence diagrams are an excellent way to document usage scenarios and to both capture required object early in analysis and to verify 'object usage later in design. Sequence diagrams show the flow of message~ from one object to another, and as such correspond to the methods and events supported by a class/object.

Figure 10.11 shows an example of a sequence diagram, with the user or actor on the left initiating a flow of events and messages that correspond to the Use Case scenario. The messages that pass between objects will become class operations in the final model.



4.0 CONCLUSION

In this unit, you have learnt about the visual and emerging CASE tools. You would also have learned about the features of traditional and

CASE-based system development. Finally, you would have understand the classification of CASE tools..

5.0 SUMMARY

What you have learned in this unit concerns CASE tools, for systems development.

SELF ASSESSMENT EXERCISE

1. List the features found across object-oriented CASE tools.
2. Discuss sequence diagrams.

6.0 TUTOR-MARKED ASSIGNMENT

What distinguishes object oriented CASE tools from others?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 5 IMPLEMENTATION OF SYSTEMS

CONTENTS

1.0 INTRODUCTION

Conducting system tests, preparing conversion plan, installing database, training end users, preparing user manual and converting to the new system are all considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

conduct system tests
prepare conversion plans
install Database
train the end users
prepare user manual
move to the new system.

3.0 MAIN CONTENT

3.1 Implementation of Systems

Implementation of system involves developing working computer software from the design specification through coding by a team of programmers. Many times, the user requirements are either not built-into the design specification or compromised to make the design simple and manageable. Implementation of the system is done by coding, testing and creating the necessary hardware and network environment, and imparts training to the end users. Of course, apart from Coding and Testing, the running implementation activities differ from project to project. This phase of the software development requires intensive user involvement.

3.6 Conducting System Tests

No system can be perfect. Testing is of vital importance as all information systems are designed by a team of Software Engineers and end users have little or no knowledge of system development. Testing is done to bridge the gap between the perceived out comes desired by the user to that of systems analysts and programming team. The design specifications are requirements of the user and are translated to working software by the programmer. Hence; it is the ability of the programmer to code exactly as per the design specification that is to be judged by testing the software module.

The objective of any testing mechanism is to discover and fix bugs before the product is delivered to the customer. A good testing scheme has a high probability of discovering an undiscovered error. The objective of any good testing scheme is to find and fix bugs with minimum time and resources. Besides, bugs and errors systems are tested for response time, volume of transactions that can be handled, stress under which it can function, security and usability. For an Online Transaction Processing System, testing of the system for response time could be quite vital.

System testing assumes that all parts of the system are correct and error-free. Even though the system has been tested for individual components and modules, there is no guarantee that the system after integration will work as per the desired specification. System test involves a holistic approach for testing the working of the application in totality.

The following are various types of System Testing:

Recovery Testing: Test the ability of the system to recover from errors. Errors or any other processing faults must not cause overall system to fail. The recovery time of the system after failure must be within a specific period and tolerance limit. System failures are forced during this phase of testing by introducing exceptions to see how the system responds to the case.

Security Testing: System used for processing sensitive information are prone to high security risks. Individual often tries to access unauthorized data for various reasons. Threats could be external or internal. Hacking of passwords is a common problem. Individual can use software to generate random passwords to gain access of the system. Security testing takes care of these aspects of the system security.

Stress Testing: Stress test is designed to test the system as to how the system behaves in abnormal situation. The aim of the stress test is to find the limit of quantity or frequency of input after which the system fails. Stress test cases are designed which require maximum memory and other resources; in excess of what a normal situation demands.

Performance Testing: Performance testing is specifically important to embedded and real time systems. It checks the run time performance of the system. It is often coupled with stress testing.

Response Testing: Testing of response time is of special importance in OLTP (on- line transaction processing systems like railway reservation system, points of sale, etc.). Testing is done to measure the response time. The same is compared with desired maximum response time.

Usability and Documentation Testing: Testing is done to review the usability and user friendliness of the software. Most often, systems are provided with on-line help screen to help the end user. This also includes whether proper care had been taken to document the development stage of the project. User friendliness of the system is often compromised, which may lead to problem during implementation and maintenance of the system.

The following are the various activities involved during system testing:

Preparation of Test Plan: The first step in system testing is to prepare a document called a Test Plan. Test plan is a document which outlines the aspect of the system to be tested. A workable Test Plan is prepared in accordance with the design specification such as:

Expected output from the system;
Criteria for evaluating the output;
Nature and Volume of test data; and
Procedure for using the test data.

Specifications of User Acceptance Test: User is involved to prepare test cases. These can be derived from the test plan. Other parameters are test schedule, test duration and the person delegated to carry out the user acceptance test.

Preparation of Test Data for Testing: Test data are often generated during testing of program. The test data must be true representative of the live data to be actually used - by the end users after installation. Care should be taken to select the nature and volume of data.

Although enough care is taken to test the system as per the documented specification, it is almost always a confusion regarding how the user will use the end product. In case there is one customer (a specific application designed for a specific use), a series of acceptance tests are carried out to validate all the user requirements. But this is not possible if the software is to be used by many customers (general purpose software like word processor, etc.). An alternate approach is application of Alpha and Beta testing techniques.

Alpha Testing: Alpha testing is carried out by the customer at the developers' site. The customer uses the software and records the errors/bugs and usage problem. Alpha testing is carried out in a controlled environment.

Beta Testing: Beta testing is carried out at one or more customer sites by the end users. It is live testing of the software product and not controlled by the developer.
The customer tests the software using her/his own data records and reports the bugs or problems in regular intervals to the developer.

Many organizations deploy specially trained personnel for system testing. The problems of bugs uncovered in alpha and beta testing are fixed before the product is shipped and installed in customer's premises. Testing of complex software can be time consuming and frustrating also. The aim of system testing is to uncover every possible error that may come up at the user end. The role of Data Processing Auditor (EDP auditors) information System Auditor is quite involved during all stages of system development especially during testing. Auditors can provide useful independent inputs to minimize complications during maintenance.

3.7 Preparing Conversion Plan

A conversion plan is a document which spells out detailed requirements for a successful conversion from existing system to proposed system. The complexity of conversion is directly proportional to the complexity of the system in question. An important role of Systems Analyst is to see that the newly designed system is implemented to the set specification. Conversion is just one aspect of implementation, other being software maintenance and system review.

A proper conversion plan ensures that conversion from old system to new system is smooth without affecting the normal business operation. The conversion process can be tedious and disrupt normal functioning of system and also involves financial and human resources. A well designed conversion plan facilitates a smooth switch over to the new system while keeping the cost and human involvement to the minimum.

A typical conversion plan is a document that consists of the following information:

Guidelines regarding Conversion processes involved and the roles of end user.
Planning conversion of files, creation of computer compatible files.
Types of conversion to be undertaken depending on the existing types of system. It could be from an existing manual system to a newly designed system or from an existing old computerized system to a newly designed enhanced system.
Types of conversion may be parallel, phased or direct.
Evaluation of hardware, software and related services.

3.8 Installing the End User

Installation of database is nothing but creating computer readable files from the existing systems/documents. Each installation involves data. The new system is going to use data created either manually or data that has been obtained from the old system. If the current system is using computer readable data, it must be made error free and compatible for use in the new system. The data must be converted to the new format supported by the current technology on which the system is being developed.

Usually, there will be upward compatibility between various versions of software. The data conversion process can be tedious depending on the format supported by the new system. Special software are designed to facilitate the installation of Database.

3.9 Training the End User

Training the user is one of the vital activities. The project team must make sure that the end users are trained to operate the new system. Many systems fail to get implemented or deliver the desired result because the end users are not trained.

Managers and the users must be trained on fundamentals of information technology in addition to knowing the operation of the new system. Training and support form the two crucial issues involving success of any information system. While training is imparted in a fixed schedule, support is an ongoing process. In support activity, the user is provided continual operational and technical support to carry out the work.

Support materials are developed to facilitate this task. The goal of any training and support activity is to achieve highest possible productivity with lowest cost. Training may involve the following activities:

Entering the data into the system. Generating the required reports.
Basic training of computers not specific to the application program like copying a file, starting and shutting down system, etc.
Briefing about Hardware and Software concepts.
Reporting non compliance and bugs in the program? Process of taking backup of daily work.

There is no exhaustive list of training requirement of the end user and can vary depending on the nature of application. The training must be scheduled in logical sequence depending on the pre-requisite for the

next module of the training. A dependency chart could be useful for this purpose.

Training can be imparted in different ways:

Computer-aided training
Classroom tutorial
Interactive training manual
Resident technical expert
One to one training
External sources
Information center / help desk

Many organizations have a well-developed automated support mechanism for end user support. To make a system success, following issues should be taken care of:

Develop a satisfactory support base for providing support to the user;
Obtain user participation and commitments;
Institutionalize the system of training; and
Insist on mandatory use of the information system.

Regular training should become part of the organization's policy as information system changes as per the requirement of the organization and new features are added.

3.10 Preparing User Manual

All information systems are unique and different from one another. Documentation starts from the day one of system development lifecycle, but preparation of end user documentation is of specific importance as the end user does not understand the intricacies of system development and hence operational problems are bound to occur. Documentation of any information system is generally of two types. System Documentation and User Documentation. System documentation contains detailed information about systems design specification, its internal structure and related technical details. The system documents are primarily for the programmer for maintenance purpose. The user documentation on the other hand is for the end user. The document should be structured and self-contained.

A user manual generally contains written as well as pictorial representation of the information system about its working and application. A well-designed user manual can reduce the overall cost of training and support. On-line help system with hyperlinks and context

sensitive help systems are slowly replacing bulky and non- interactive documents.

The following are the components of a User Manual:

Title and Version of software release
Table of contents
Salient features of the product
Installation Guide and System requirements
Getting started
Frequently asked questions
Sample scenario
Glossary of terms used in the manual
Known bugs in the applications

With changing technology, user documentation is often bundled to the information system. It is separate document. On-line documentations are being extensively utilized in user environments due to their convenience. Context sensitive helps are making the users' life easy by reducing the time to browse the bulky documents.

3.11 Converting to the New System

Actual conversion process involves equipments, personnel, data and financial resources. The process of conversion from the existing system (manual or computerized) to the newly developed system can be performed in several ways depending on the criticality of the system and other related issues.

Direct Conversion: This is abrupt approach. The old system is shutdown and the new system starts. This kind of conversion although economical, the users are at the mercy of the new system, hence direct installation can be very risky. Some times due to procedural reasons where two systems can't be run parallel, this kind of conversion is the only option. When the new system fails, there is no way to start the old system as a backup as it has been shutdown. This kind of conversion plan is often the least preferred for critical business applications.

Pilot Conversion: This is the middle path approach. Instead of converting all at once throughout the organization, this kind of pilot installation involves conversion/installation of system at a single pre-decided location. The location may be a branch office of the organization. Proper selection of the pilot site is important as it should be able to perform a true conversion process to test all functionalities of the new system. The advantage of the pilot

conversion is that the potential risk in case of failure of the system is limited to a single location. Once the user is ascertained that the implementation of the system has been successful in a particular location, it is proposed to replicate the system in other locations. Although this kind of pilot conversion plan is beneficial for the user, it places a substantial burden on the implementation team as it has to maintain two systems in parallel.

Parallel Conversion: is least risk prone. Under this kind of conversion, the old system is allowed to run alongside the new system until the management and the end user are satisfied with the result of the n6W system. It is compared with the new system to test whether the functionalities covered by the old system are thoroughly covered in the new system by comparing the outputs. Errors and bugs identified with the new system are not detrimental for normal functioning of the organization as the new system is replaced and normal functions are resumed by the old system. Parallel conversion is costly as two systems are run in parallel, but results of only one system are used for business operations.

Phased Conversion: is an incremental approach to switch over to the new system. Different sub-systems of the new system is used in conjunction until the whole new system is converted. This kind of approach for conversion limits the potential risk of failure of the new system. In a phased installation as a sub-system is made functional, actual results are visible before the whole new system is made functional.

Each conversion strategy not only involves data and software, but also other resources like personnel, hardware, etc. Hardware and software selection is an important issue to be considered before actually carrying the conversion.

SELF ASSESSMENT EXERCISE

1. Describe any two types of system testing.
2. What are the components of a user manual?

4.0 CONCLUSION

In this unit you would have learned to conduct system tests, prepare conversion plans, install database, train end users, prepare user manuals and move to the new system.

5.0 SUMMARY

What you have learned in this unit pertains to the implementation of system.

6.0 TUTOR-MARKED ASSIGNMENT

What is the objective of any testing mechanism?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 6 MAINTENANCE OF SYSTEMS

CONTENTS

1.0 INTRODUCTION

System maintenance involves more than 80% of the total life of a software product. Different maintenance activities as well as issues involved in maintenance are all considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

be aware of the key stages in any maintenance activity
understand the different maintenance activities
describe issues in software maintenance.

3.0 MAIN CONTENT

3.1 Maintenance of Systems

Once the information system is successfully installed and started showing result, the next issue is to maintain the system. System maintenance involves more than 80% of the total life of a software product; this shows the importance of maintenance. System maintenance is the task of monitoring, evaluating and modifying the information system to make necessary desirable changes during the total life cycle of the software. Organizational requirements as perceived during the analysis phase changes, the system has to accommodate all such changes to make the system current and useful for the organization. Maintenance of system also takes care of the failure and shortcomings that arise during the operation of the information system by the end user. During the implementation phase, one person from the system maintenance group is nominated to collect information from the user for maintenances. Maintenance activity involves collecting requests for changes, transforming these requests to changes, designing the changes to be incorporated and implementing the changes in the system.

Any maintenance activity comprises the following four key stages:

Help Desk: The problem is received from the user through a formal change request. A preliminary analysis of the change request will be done, and if the problem is sensible, it is accepted.

Analysis: Managerial and technical analyses of the problems are undertaken to investigate the cost factors and other alternative solutions. Feasibility Analysis is done to assess the impact of the modification, to investigate alternative solutions, to assess short and long term costs, and to compute the benefit of making the change.

Implementation: The chosen change/solution is implemented and tested by the maintenance team. All infected components are to be identified and brought in to the scope of the change. Unit test, integration test, user-oriented functional acceptance tests and regression test strategies are provided.

Release: The changes are released to the customer, with a release note and appropriate documentation giving details of the changes.

3.6 Different Maintenance Activities

Once the system is fully implemented and starts operating, the maintenance phase begins. When the user starts operating the system, initial difficulty diminishes as the user learns to operate the system. The maintenance may include modification of system due to changes in business environment, government regulations, new business ventures and enhancement of functionalities.

The majority of Software Maintenance activity is concerned with evolution derived from user requested changes. A program that is used in a real world environment necessarily must change or become progressively less useful in that environment. As an evolving program changes, it's structure tends to become more complex. Extra resources must be devoted to preserving the semantics and simplifying the structure.

The following are the different types of maintenance activities:

Corrective Maintenance: This type of maintenance is to rectify design, coding and implementation problems detected after the implementation of the System. This kind of problem generally surfaces immediately after the system is implemented. This type of problem needs immediate attention as it hampers the day to day work of the end user. Proper planning and interaction with the end user during system development process can minimize Corrective Maintenance. In spite of the all these kinds of maintenance, these constitute more than 60 percent of total maintenance effort. Corrective maintenance is very much undesirable. It does not do any value addition to the software. Care should be taken to see that normal business operations are *not* disturbed because of it.

Adaptive Maintenance: Changes are needed as a consequence of upgraded versions or changes in operation system, hardware, or DBMS. Adaptive maintenance is required because business operates on a social environment and need of the organization changes as organization ventures in to new areas, or as government regulation policy changes, etc. Maintenance of the software to adapt to this kind of changes is called adaptive maintenance. Unlike corrective

maintenance, this kind of activity adds value to the information system and affects a small part of the organization. This activity is not as urgent as corrective maintenance as these changes are gradual and allow sufficient time to the system group to make changes to the software.

Perfective Maintenance: This kind of maintenance activity involves adding new functionalities and features to the software to make it more versatile and user oriented. Some times, changes are made to improve performance of the software. In some sense, this maintenance can be thought of as a new development activity. This adds value to the information system and is required to stay ahead of the competition.

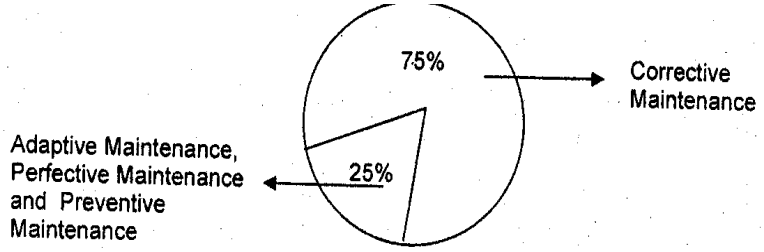


Figure 11.1: Comparative figures of Maintenance effect

Preventive Maintenance: Changes are made to software to make it easily maintainable and to prevent any kind of system failure in future. This reduces the need of corrective maintenance. As corrective maintenance could lead to hamper normal functioning, preventive maintenance is done periodically to ensure that the probability of system failure is minimized. Preventive maintenance could increase the volume of transactions that can be handled by the system. Preventive maintenance is done when the system is least used or not used at all. This does not add value to the system, but certainly lowers the cost of corrective maintenance.

Figure 11.1 depicts the maintenance efforts that are to be put during each maintenance.

3.7 Issues Involved in Maintenance

The responsibility of the software development team and clients does not end once the product is released for implementation and installed. If software is not properly maintained, a well-documented and cleanly designed system can decay into a poorly documented and ill-maintained system. Additional venerability may get introduced luring the activity of maintenance. In a network environment, a bug has ramification beyond just poor performance or functionalities. A bug can open up avenue for a hostile intruder.

It is very important that the Software should be easily maintainable. Factors like availability of source code, availability of system manuals, etc., are very important or maintainability. One of the most important issues is the cost factor for maintenance of software. There are a number of factors that influence the cost of maintenance. Maintenance activity may some times introduce new bugs while rectifying it.

The following are various factors which affect the ease of maintenance:

Volume of Defects: The inherent errors / bugs that are found in the system after installation. Cost of maintenance increases with the increase in volume of defects.

Number of Customers: More number of customers means more requests for changes in the system after installation.

Availability of System Documentation: The quality and availability of system documentation is vital to carry out the maintenance. Poorly written system documentation increases the cost of maintenance. Most often, the programmers for development are different than the team of programmers for maintenance and the later often finds it difficult to understand a program written by the former. Structured programming and program documentations are very useful in maintaining the system.

The following are various issues in Software Maintenance:

1. Organizational Issues

The maintenance activity must align with organizational objectives.
Most of the Software Maintenance activity is resource consuming and it has no clear quantifiable benefit for the organization.
Outsourcing the job of Software Maintenance

2. Process Issues

Software Maintenance requires a number of additional activities not performed at development stage. Impact analysis and Regression tests on the software changes are crucial issues

3. Technical Issues

How to construct software that it is easy to comprehend is a major issue and the technology to do this is still not available. Still, the following are some guidelines for the same:

Translate the problem into software terms to decide if it is viable or not. Determine the origin of the change request and suggest solutions.
All solutions are investigated to determine that they are applied to all software components affected.
Make a decision on the best implementation route or to make no change.
Ripple effect propagation is a phenomenon by which changes made to a software component along the software life cycle have a tendency to be felt in other components

Legacy System

A legacy system is typically a very old and large system which has been modified heavily since it started operation. Legacy systems are based on old technology with very little or no documentation. Dealing with a legacy system can be very hard.

Solutions for the problems mentioned above relating to a Legacy System:

Explore possibility of subcontracting the maintenance
Replace software with a package
Re-implement from scratch
Discard software and discontinue
Freeze maintenance and phase in new system
Reverse engineer the legacy system and develop a new software suite.

SELF ASSESSMENT EXERCISE

1. List 3 factors which affect the ease of maintenance
2. Issues in software maintenance involve process issues. Discuss

4.0 CONCLUSION

Out of all the maintenance activity, corrective maintenance constitutes more high percent of the total maintenance activity. Different issues related to software maintenance were also considered in this unit.

5.0 SUMMARY

What you have learned in this unit borders on the maintenance of systems.

6.0 TUTOR-MARKED ASSIGNMENT

Why is corrective maintenance performed?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com
http://www.dur.ac.uk/csm/jsm

UNIT 7 AUDIT OF COMPUTER SYSTEM

CONTENTS

1.0 INTRODUCTION

In this unit you will learn about the definition of audit, audit of transaction on computer and computer assisted audit techniques.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

define audit
list the objectives of audit

explain the responsibility of the system auditor
describe the computer assisted audit techniques.

3.0 MAIN CONTENT

3.1 Definition of Audit

This is an assessment of an information system performed by an information system professional or IS auditor to provide recommendations and advice to improve system performance aria security. Audit should be done regularly and the result should be used to refine the system.

Is auditors are those people who make it sure that the system does what it is supposed to do. Although the audit can be carried out by the internal team of IT professionals, it is advisable that the audit is carried out by external auditors as they are neither stakeholders nor friendly with the stakeholders; Above all there is nothing like an unbiased opinion.

Information System auditor is a person who engages in Information system audits with the following knowledge and abilities:

1. Basic knowledge of information systems.
2. Knowledge of system audits.
3. Ability to perform system audits.
4 Objectives of Audit

A. Objectives of Audit

The following are the objectives of Audit:

To improve the quality of information systems, prevent failure and minimize the effects of failure, and speed up the process of recovery in the event of a failure. This will help Information System to be more reliable.
To make an information system more secure from natural as well as manmade disasters, unauthorized access, and other destructive actions.
To improve the cost performance of an information system by optimum utilization of its resources, which leads to increase in efficiency?

During the course of audit, the Information Systems Auditor will obtain sufficient, reliable, relevant and useful evidence to achieve the audit objectives effectively. The audit findings and conclusions are to be supported by appropriate analysis and interpretation of this evidence.

To achieve the above objective, the following documents should be made available to the auditors A diagram of the Information System (Application)

1. Network diagram
2. A hierarchical diagram of the project team

B. Responsibility and Authority of the System Auditor

The system auditor shall make the basis for each of his or her assessment clear. The system auditor may demand data and materials from the division being audited. The system auditor may also demand the head of an organization to issue a report on the implementation of improvement to an audited division as suggested by him.

The system auditor shall firmly maintain professional ethics as an impartial evaluator. The system auditor shall be aware of the ethical demands on himself or herself and meet the .internal and external trust by performing an accurate and sincere system audit;

C. Confidentiality

The system auditor with strict adherence to professional ethics must maintain confidentiality of the information provided to him to carryout his or her activity and should not, without sufficient reason, divulge any information that is classified as confidential information by the audited organization.

D. Audit Planning

The Information Systems Auditor has to plan the information systems audit Work to address the audit objectives and must comply with applicable professional auditing standards.

3.4 Audit of Transactions on Computer

Audit can be broadly of two types namely auditing manual processes and audit through computer. Audit through computer is important to find out the accuracy and integrity of information system output. This types of audit are done by information system expert and use test data to check the adequacy and accuracy of control mechanism built-in to the system.

A typical audit looks at the following factors:

Audit of Response Time: In this audit the actual response time of the system verses the desired response time is compared to the performance of the system

Audit of Broken Links: This is applicable to web site and other intranet applications. The most irritating things on a web site is not finding a link document. There are automated software to find broken/unavailable links on web site.

Database Audit: Database audits involve checking the database integrity and availability. The information that is sent to the database should be checked with the information actually stored on the database.

Network Audit: Network audit involves checking the venerability of network. It checks whether the network configuration is giving optimal performance or not.

A. Transaction Audit

Transaction audit is a process to find:

Who did changes?
What changes are made?
Whether the changes are authorized or not as per the security policy of the Organization?

The details of the above transactions are written to either a media or printed. This allows Database Administrators to track changes and helps the organization to satisfy regulatory requirements such as tracking specific users actions, general security screening, validating user permissions etc.

B. Audit of Computer Security

Issues of security of computer involve both physical and logical security. Physical security involves restricting physical access to the computing resources from unauthorized person. Logical security involves restricting the use of computing resources by unauthorized person by providing logical control mechanism (e.g. password protection). The audit of computer security involves review of physical and logical security measures. Review of parameters, plans, practices, and policies that are developed and implemented by the organization over the computer resources, and how security measures are followed for Computers, Networks and Data communication. They are also included in the Audit.

C. Audit of Application

Here, both manual and programmed internal controls related to information systems are assessed. Primarily, there are four areas of audit coverage for an application being reviewed.

The four areas are given below:

Control Environment: This includes reviewing the system's security, its operating platform, system documentation and the interaction it has with other systems.

Data Input Controls: This involves reviewing the controls which ensure that data that enters into the system is accurate, complete and valid as per the standard.
Examples include verifying system tables, limit checks, range checks and redundant data checks.
Processing Controls: These controls ensure that the data is properly processed and that automatic calculations performed by the system are accurate. This is tested by assessing controls built into the programs and by processing test data through the system and comparing the results of processing with expected results. Also, there will be checks on currency of stored data, default values and reporting exceptions.

Output Controls: In this, review of the system generated reports to ensure that they are accurate and the reports produced are reliable, timely and relevant is done. Also, it is checked whether cost savings can be achieved by reducing the number of reports produced. Data control personnel perform visual review of computer output and reconciliation of totals.

D. Benefits of Audit

Information system audit is increasingly becoming the focal point of the independent audit, compliance audit, and operational audits. An information system audit can help the organizations in many ways:

Improve system and process controls.
Prevent and detect errors as well as fraud.
Reduce risk and enhance system security.
Plan for contingencies and disaster recovery.
Manage information & developing systems.
Prepare for the independent audit.
Evaluating the effectiveness and efficiency related to the use of resources.

Cost control.
Competitive advantage

3.5 Computer Assisted Audit Techniques

The auditors use various types of automated audit software to carryout IS audit. The use of Computer Assisted Audit Tools (CAATs) should be controlled by the IS. Auditor to provide reasonable assurance that the audit objectives and the detailed specifications of the CAATs have been met. There are two major types of CAATs namely *audit software* and *test data.*

Audit Software

This is a computer program used to process data of significance for audit from entity's accounting system. The auditor should substantiate their validity for audit purposes before making use of these tools. These include:

a) Package programs: Generalized computer programs to perform data processing functions like reading computer files, selecting info, performing calculations, etc.
b) Special purpose programs: Computer programs designed to perform audit tasks in specific business circumstances. .
c) Utility tools: Used by the auditors to perform common data processing functions like sorting, creating and printing files. These tools are not designed for audit purposes specifically.

Various commercial Audit Software are available to carry out system Audit. Some of them are:

1. Visual Audit Pro
2. IDEA
3. E-Z Audit

Visual Audit Pro: It audits automatically over a network. It audits activities like, use log on/off, collects information about software and its version, collects information about hardware inventory like serial number, model, memory and associated peripheral devices, user information, registry information etc.

E-Z Audit: With this software one can know information on capacity of RAM, name of network card with its connect speed, MAC address and TCP/IP information. You can also find out how many local, removable and network drives are there on the system, what printers are connected, both networked and local, etc. On software front, it gives information on

name and version of OS running on the system with service packs, installed programs and their names, EXE files and DLL version.

IDEA (Interactive Data Extraction and Analysis): IDEA can be used to import information from database to be audited for further analysis to auditor. It helps to corroborate audit evidence effectively. For example it can check for duplicate payment on a single invoice. It is useful to analyze system log for fraud detection.

Consider the audit of a Payroll Package. The potential fraud that can occur in a payroll system is very high. Therefore, audit software is used as detection tool for fraud. The Audit software looks for salary unusually high, extracting information without a department number, extract information on bank account number. It also can extract information on fictitious employee, compare it with personnel database. It can also compare payment details of two different months.

Test Data

Test data is used to test the correctness of the software. When test data is processed with the entity's normal processing systems, the auditors should ensure that the test transactions are subsequently eliminated from the system. When using the test data, the IS auditors should be aware that the test data should only point out the erroneous processing a¥ should not change the data that is produced by the system during real life.

Audit Expert Systems

Some IS auditors make use of Expert Systems to assist in auditing. When using these audit expert systems, the IS Auditor should be thoroughly knowledgeable of the operations of the system to confirm that the decision paths followed are appropriate to the given audit environment or situation.

Audit Trail

Audit trail is a log of changes made in the data, settings and related changes. A security subsystem should maintain detailed logs of who did what and when and also if there are any attempted security violations. The availability of the log is extremely valuable. Log provides information for the system auditor to be able to determine who initiated the transaction, the time of the day, date of entry, the type of entry, fields of information that were affected and the terminal used.

System log should be analyzed to provide detailed information on all normal and abnormal transactions during each processing period. System access and attempted access violations can be automatically logged by the computer and can be reported for check & review. Listing of terminal addresses and locations can be used to look for incorrectly logged, missing or additional terminals.

Applying the principles of Information System Security and Audit raised in this write-up will ensure that an organization's information assets and systems are adequately controlled, monitored and assessed.

SELF ASSESSMENT EXERCISE

1. Define the term 'Audit'.
2. List three computer assisted audit techniques.

4.0 CONCLUSION

You would have learned how to define audit, list the objectives of audit, explain the responsibility of the System Auditor and the computer assisted audit techniques.

5.0 SUMMARY

You have learned about the audit of computer systems in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

What are the objectives of Audit?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 8 SEURITY OF COMPUTER SYSTEMS

CONTENTS

1.0 INTRODUCTION

In this unit we will learn about the analysis of threats and risk, how to recover from disaster, how to plan the contingencies, as well as concurrent audit techniques.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

describe the analysis of threats and risks
plan the contingencies in the event of disasters
protect information systems from virus.

3.0 MAIN CONTENT

3.1 Computer System and Security Issues

Security is an important issue for modem IT Systems. Even though technology provides immense possibilities *to* safeguard organizations computing infrastructure there has been security lapses and security breaches which have cost the organization heavily. System administrator and security administrator have spent sleepless nights to safeguard organization's data and computing infrastructure. One can think of organization like airlines, railway and banks which are heavily dependent on computing infrastructure and unavailability of system for few hours can create havoc.
Organizations can't afford to underestimate the security issues that can affect their business operations.

There may be security threats due to natural reasons such as Earth Quakes, Cyclones etc. Sometimes, the threats are made by people. These may be due to riots, unrest, sabotage etc. Whenever, there is an attack, immediate reactive measures are to be taken. Also, one should study various controls to find out the people or reasons behind the attack. This can be done with the help of transaction logs etc. These attacks basically become possible due to several drawbacks in the information system such as lack of proper implementation of security protocols etc. Such things are exploited by people who plan attacks. The entire situation surrounding attacks is depicted in Fig. 12.1.

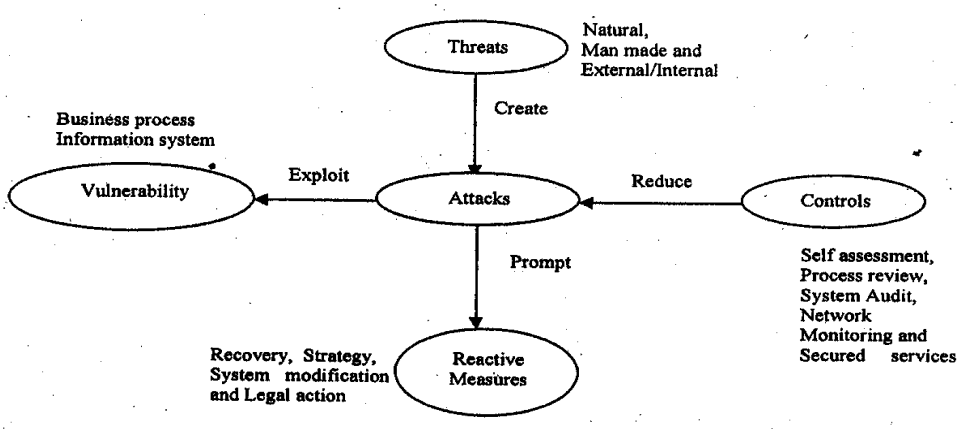


Figure 12.1: Information Security Architecture

3.4 Analysis of Threats and Risks

The security of any system should be commensurate with the risk involved. Threat and risk assessment involves identification of applicable threats to IS infrastructure, recognition of vulnerability and probable loss calculation. In this context, it is necessary to identify the source of threat.

Historically, an organization's computer systems were centrally located and the management of issues related to it were responsibility of the computer center staff and as such security related issues were also the responsibility of computer center staff whose focus were to make available the application on the centrally located computer as required. In comparison, today's computing infrastructures are far more diverse and complex to manage. Business information is dispersed.

The source of threats can be either external or internal. Historically viruses has been the major potential external security threat but as organizations are diversifying their activity over multiple locations and with evolution of new technology it is difficult to perceive when an unauthorized intruder may try to hack upon organization's vital information and cause damage. Internal security threats are more common although the integrity of employee is checked before being inducted into the organization. Employee of an organization can pose serious threats to information security as they are closely associated with the system and know the vulnerabilities that can be targeted.

Risk Analysis

The common questions asked in evaluating the risks are given below.

Are the risks such as fire, earthquakes and the scope of their effects on the information system been made clear?
Has the loss, the organization would suffer from a halt or the like of the information system been analyzed?
Is the time permissible for recovery of operation and the order of priority of recovery been determined?

Security policy underlines an organization's holistic approach to security issues. Organizations must possess a security policy in writing, which should address the following issues:

Authentication: To see that the person is bonafide user of the resources
Authorization: Privileges of the user or who can do what?

Information integrity: Is it possible that the end user can modify the information? **Detection**: Once the problem is identified, how it is handled and managed.

Risk Assessment and Management

A thorough and proactive risk assessment is the first step 1n establishing a sound security system. This is the ongoing process of evaluating threats and vulnerabilities, and establishing an appropriate risk management program to mitigate potential monetary losses and harm to an institution's reputation. Threats have the potential to harm an institution, while vulnerabilities are weaknesses that can be exploited.

There are different approaches followed by organizations to analyze risks. However, ultimately all the methods boil down to two types of approaches: quantitative and qualitative.

Quantitative Risk Analysis

This approach although difficult to implement gives an idea about the amount of risk involved with the event. This basically employs two fundamental elements i.e. the probability of occurrence of the loss making event and probability of occurrence of the event.

*Estimated Loss = Potential loss due to the event * probability*

It is therefore possible to rank the events in order of estimated loss. But the problem associated with the quantitative approach is estimating the probability of occurrence of the event, also in some cases the events are interrelated making the probability calculation even more difficult. Notwithstanding above difficulty, many organizations have adopted and implemented this approach successfully.

Qualitative Risk Analysis

The extent of the information security program should commensurate with the degree of risk associated with the institution's systems, networks, and information assets. For example, compared to an information-only Web site, institutions offering transactional Internet banking activities are exposed to greater risks. Further, real-time funds transfers generally pose greater risks than delayed or batch-processed transactions because the items are processed immediately. The extent to which an institution contracts, with third-party vendors will also affect the way the risk assessment has to be done.

Performing the Risk Assessment and Determining Vulnerabilities

Performing a sound risk assessment is critical to establishing an effective information security program. The risk assessment provides a framework for establishing policy guidelines and identifying the risk assessment tools and practices that may be appropriate for an institution. Banks still should have a written information security policy, sound security policy guidelines, and well-designed system architecture, as well as provide for physical security, employee education, and testing, as part of an effective program.

When institutions contract with third-party providers for information system services, they should have a concrete opinion about third party provider's quality of work and loyalty to the clients. At the minimum; the security-related clauses of a written contract should define the responsibilities of both parties with respect to data confidentiality, system security, and notification procedures in the event of data or system compromise. The institution needs to conduct a comprehensive analysis of the provider's security program, including how the provider uses available risk assessment tools and practices. Institutions also should obtain copies of independent penetration tests run against the provider’s system.

When assessing information security products, management should be aware that many products offer a combination of risk assessment features, and can cover single or multiple operating systems. Several organizations provide independent assessments and certifications of the adequacy of computer security products (e.g., firewalls).
While the underlying product may be certified, banks should realize that the manner in which the products are configured and ultimately used is an integral part of the products' effectiveness. If relying on the certification, banks should understand the certification process used by the organization certifying the security product. Other examples of items to consider in the risk assessment process include:

Identifying mission-critical information systems, and determining the effectiveness of current information security programs. For example, vulnerability might involve critical systems that are not reasonably isolated from the Internet and external access via modem. Up-to-date inventory listings of hardware, software, as well as network topologies, is important in this process.

Assessing the importance and business sensitivity of information for the likelihood of outside attacks/hacking and internal misuse of information. For example organization could be harmed if human resource data (e.g., confidential personnel information) were made

public. The assessment process should identify systems that review the appropriateness of access controls and other security policy settings.

Assessing the risks posed by service provider or business partner through electronic connections with internal IT infrastructure. The outsider may have poor access controls that could potentially lead to an indirect compromise of the organizations security system.

Determining legal implications of security breaks and contingent liability concerns associated with any of the above factor. For example, if hackers successfully access a bank's system and withdraw money fraudulently, the bank will be liable for damage incurred to the account holder.

Potential Threats

Denial of service (DoS), which can be described as any action that prevent a system from normal operation. It may be the unauthorized destruction, modification, or delay of service. DoS is common where the number of requests outnumber the maximum number of connections possible. Under such circumstances, legitimate users have to wait for large amount of time for response to their request.

Internet Protocol (IP) spoofing, which allows an intruder via the Internet/intranet to effectively impersonate a local system's IP address in an attempt to gain access to the system. The system in this case may misinterpret the incoming connection as originating from a trusted host

A Trojan horse program generally performs unintended destructive functions that may include destroying data, collecting invalid or falsifying data. Trojan horses can be attached to e-mails.

Viruses are computer programs that may be embedded in other program and have the capability to self-replicate. Once active, they may result in either nondestructive or destructive invalid outcomes in the host computer. The virus program may also move into multiple platforms, data files, or devices on a system and spread through multiple systems in a network or through emails to other systems.

3.5 Recovering from Disasters

Natural and man-made disasters are inevitable. Earthquake, floods, fire and terrorist attack can severely damage organizations computing infrastructure. The disaster recovery plan is a document containing procedures for emergency response, extended backup operations, and recovery should a computer installation experience a partial or total loss of computing resources or physical facilities (or of access to such facilities). The primary objective of this plan, used in conjunction with the contingency plans, is to provide reasonable assurance that a computing installation can recover from disasters, continue to process critical applications in a degraded mode, and return to a normal mode of operation within a reasonable time. A key part of disaster recovery planning is to provide for processing at an alternative site during the time that the original facility is unavailable.

Contingency and emergency plans establish recovery procedures that address specific threats. These plans help prevent minor incidents from escalating into disasters. For example, a contingency plan might provide a set of procedures that define the condition and response required to return a computing capability to nominal operation. An emergency plan might be a specific procedure for shutting down equipment in the event of a fire or for evacuating a facility in the event of an earthquake.

During a disaster, normal operating procedures may be significantly altered. Both personnel and systems will be expected to function under conditions that are not expected under normal day-to-day operations. Security remains a requirement but techniques to apply it are altered to fit the contingency situation.

In-House Backup

This level is the minimum acceptable and is mandatory for all installations and application's systems. Define in detail all in-house back up procedures, the techniques used, files copied, frequency, etc.

Alternate Storage Area

This level of protection is necessary for mission critical components. It consists offsite storage of at least one copy of all AIS files and databases, programs, and procedures necessary to operate the high priority application systems, either at the installation or at an alternate site of operation (including copies of contingency plans and related materials).

The alternate storage area should be located in an area reasonably accessible to the installation, but not subject to the same degree of major threat as the site. It is recommended that, as a rule of thumb, the

alternate storage area be no closer than one mile from the site. However, the distance may vary from location to location.

The Disaster Recovery Toolkit

The *Disaster Recovery Toolkit* is a highly valuable collection of items and documents to assist in ensuring business continuity in the face of serious incident or disaster. Many organizations use these documents as a checklist and add element specific to their need.

Although they vary from organization to organization, they generally comprise the following:

A contingency audit questionnaire.
A dependency analysis document - questions and guidance.
A Business Impact Analysis questionnaire.
An audit questionnaire for disaster recovery or business continuity plan.
A checklist, action list and framework for disaster recovery

The toolkit is designed to help review the full spectrum of business continuity and disaster recovery issues.

3.3.1 Planning the Contingencies

Every business entity can and do experience events which Call prevent it from normal function. The factors can range from natural events like flood, fire, earthquake etc. or a man made events like unauthorized access, serious computer malfunction or various information security accidents.

The very first step for contingency planning is to identify the contingency events covered and the appropriate actions for each. Contingency events usually refer to varying degrees of loss across six major asset categories: Data, Software, Communications, Hardware, Personnel, and Facility. The cause of the loss is dealt with in the Risk assessment, the primary concern in the contingency plan is the degree of loss, impact on the mission and techniques for coping.

Contingency management tools address basic issues such as asset identification, location, value, alternatives, replacement, and intangible costs; and most importantly, how long can the organization function without the asset? Since no asset is impervious to loss, the prudent leader will ensure that mechanisms are in place for a secure & rapid recovery. Our intent is to help managers break the cycle from normality to panic with crisis management.

Contingency Events

Loss of Data: To identify key data and the type or degree of loss/damage that would be required for necessary recovery action. It can be done as follows:

Identify appropriate recovery plan and procedure procedures. (Example in- house backups, etc.)
The location of the required recovery files.
To identify procedures for recovery of the files indicated above and include them in the contingency plan.

Loss of Software: To identify key software and the degree of criticality for necessary recovery action. It can be done as follows:

Identify the type of software (commercial / in-house developed.)
Identify the location where backup copies are maintained.
In case of an emergency procurement process, the authorized person for it and any alternate source from where operational copies can be obtained.

Loss of Communications: To identify voice as well as data communications loss for necessary contingency plan and recovery action. It can be done as follows:

Identify alternate communication facility available such as radios links or mobile phones for interim measures.
Whether there is any service agreement in place with any party to deal with contingency issues.
To estimate recovery time.

Loss of Hardware: Inventory of required hardware must be maintained. For each hardware component, the loss which would require implementation of the contingency plan has to be found. It can be done as follows:

Identify the hardware component and what functionalities it supports.
Identify any alternate piece of equipment that may be used as a substitute to the equipment and it’s degree of compatibility with the existing software.
Whether the equipment is repairable? and if so, is there maintenance agreement in place to accomplish the repairs. What is the response time to repair the equipment?

The estimated cost and time for procurement of replacement hardware in case it is not repairable.
Whether there are any emergency procurement procedures for key items?

Loss of Personnel: Loss of Personnel can result from employee leaving the organization, illness, death, family emergency and a number of other events. The following steps can be taken to minimize this type of loss:

To identify key personnel in the organization and what their involvement/impact on major systems/programs/components.

To identify substitutes for each personnel to handle such situation.

Whether there are written procedures for every important function accomplished by the key personnel. Whether the substitutes use the same procedures periodically and do the assigned tasks.

If alternates are not available within the organization replacements must be obtained from outside sources. It must be ensured that there are sufficient procedures in place and establish a training/orientation program to assign them the desired work to them.

Loss of the Facility: The loss of facility in general is due to some catastrophic natural action such as fire, flood, storm, earthquake, etc. However, a facility may become non-functional temporarily due to failure of power, or any other events that could render the facility non-functional.

If the facility is to be out of operation beyond the maximum tolerable time, identify the procedures that are necessary for moving to the alternative facility.
To identify all necessary hardware, software, data, and personnel required for normal functioning at the alternative location.
To notify the alternative location to all concerned.

Any recovery procedure generally consists of following broad steps.

Preparing contingency plan involves people from all activities. The people should understand their role in the event of disaster and should be ready' to react to the situation. Following are the major step involved in contingency planning:

Develop the Plan: The contingency plan is a detailed milestone to move the organization from a disrupted status to the status of normal

operation. The role and responsibility of each employee and service provider are defined clearly in the event of disaster.

Testing the Plan: Once the plan is ready, it should be subjected to rigorous testing and evaluation. The plan should be initially tested in a simulated environment. Persons who would actually be involved in the event of a real disaster should test the plan.

Maintaining the Plan: Once the plan is created and tested it must be kept updated so that it remains relevant and applicable to changed business environment. The changes in the business process must be reflected in the plan and all changes in it should be communicated to all concerned.

3.5 Viruses

Viruses are one of the major security threats to computer system. The first computer viruses were written in mid-eighties. The first virus written was a boot sector virus. Today, there are several tens of thousands of viruses.

Computer virus is nothing but a program that is loaded into your computer without your knowledge. This is only basic information. But, what makes people fear from Virus is the disastrous impact on remaining programs in your machine due to this program. The difference between a computer virus and other programs is that viruses are designed to self-replicate usually without the knowledge of the user. Computer viruses are called viruses because they share some of the traits of biological virus. A computer virus passes from computer to computer like a biological virus passes from person to person. A computer virus must **piggyback** on top of some other program or document in order to get executed. Once it is running, it is then able to infect other programs or documents. Obviously, the analogy between computer and biological viruses seems superficial, but, there are enough similarities as the name suggest.

Virus carries out instruction for replication. The effect of virus can vary from annoying messages, to the disastrous consequences (for example, the CIH virus, which attempts to overwrite the Flash BIOS, can cause irreparable damage to certain machines). Superficially, it looks as if virus which can format hard disk is more damaging but damage can be avoided by taking backups. Think of a virus which corrupts data by changing the numbers randomly on a spreadsheet application or changes + to -.This is certainly disastrous.

Viruses can be hidden in programs available on floppy disks or CDs, hidden in email attachments or in material downloaded from the web. If the virus has no obvious payload, a user without anti-virus software may not even be aware that a computer is infected.

A computer that has an active copy of a virus on its machine is considered infected. The way in which a virus becomes active depends on how the virus has been designed, e.g. macro viruses can become active if the user simply opens, closes or saves an infected document.

Prevention

The best way for users to protect themselves against viruses is to apply the following anti-virus measures:

Make backups of all software (including operating systems). So, if a virus attack has been made, you can retrieve safe copies of your files and software.
Inform all users that the risk of infection grows exponentially when people exchange floppy disks, download web material or open email attachments without caution.
Have anti-virus (AV) software installed and updated regularly to detect, report and disinfect viruses.
Visit sites which give information on the Internet about latest virus, it's behavior and assess their potential threat.
In case of doubt about a suspicious item that anti-virus software does not recognize, contact your anti-virus team immediately for guidance.

3.6 Concurrent Audit Techniques

Most of the Audit techniques collect data after transaction is completed. So, the outcome of the Audit is usually useful only for the future. The outcomes may be used as precautionary measures for the future.

In the case of Concurrent Audit Techniques, Data is collected while the transaction is in progress. This is very much useful for high risk transactions as they will be put on hold in case the Audit desires so. If any other Audit technique is used, then, such high risk transactions are processed after which it will be found that these transactions are invalid.

Need for Concurrent Audit Techniques

The following are few reasons for the Reed of Concurrent Audit techniques:

Missing Audit trails.
Need for continuously monitoring largely integrated and automated systems.

The following are various Concurrent Audit Techniques:

An Integrated Test Facility Technique (ITF)

In this technique, the Auditing software is embedded into the client software. Basically, what happens is that the test data of Auditor is integrated and the same is processed with Client's real life input data. ITF ensures that files of the client are unchanged and any changes, if necessary, will be made only to the dummy files of the client's files. At the end, these dummy files are studied to know the discrepancies.

The Snapshot Technique

In this technique, Audit software is embedded in the software that is to be audited. It is embedded at those places where critical processing takes place. Then, it takes a snapshot of the process before and after the critical processing.

SCARF

It stands for System Control Audit Review File. It is one of the complex Audit techniques. This technique will embed Audit software in the host application. This will enable audit software to monitor the Systems transactions uninterruptedly. The information that is collected during Audit process will be stored in a special audit file known as SCARF master file.

Usually, SCARF is used to collect the following information: Application System errors, Policy and procedural variances, System exceptions, Statistical samples, Snapshots and extended records, Data profiling, Data for performance measurement.

Continuous and Intermittent Simulation technique (CIS)

This technique will use the Data Base management systems to trap exceptions. Whenever, there is a need for service, DBMS will inform the same to CIS. CIS will then carry out the suitable service.

SELF ASSESSMENT EXERCISE

1. The major step involved in contingency planning includes?

2. Explain concurrent Audit techniques.

4.0 CONCLUSION

In this unit, you would have learned about the analysis of threats and risks, the planning of contingencies as well as how to protect the information systems from virus.

5.0 SUMMARY

You would have learned about computer system and security issues in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

What is the very first step for contingency planning?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 9 MANAGEMENT INFORMATION SYSTEM

CONTENTS

1.0 INTRODUCTION

Management Information Systems have been playing a key role in helping the managers at various levels. In this unit we will consider the role of MIS in and organization and the uses of MIS.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

explain the role of MIS in an organization
describe the application of MIS.

3.0 MAIN CONTENT

3.1 Computer System and Security Issues

Management Information System helps the organization to produce information that organizations need to improve decision-making, problem solving, control operations and creating new products or services. Many organizations have implemented computer-based

Management Information System to retain competitive edge over their competitors. The role of Management Information System (MIS) has expanded significantly over the years. Until the early I 960s, the role of MIS was simply processing transaction data, record keeping and other data processing activity. The early 1970s, evolved MIS for reporting to managers in a specified format for managerial decision-making. The early 1980s have been the time for decision support system which helps individual managers in decision-making. The early 1990's have been the age of expert system that provides knowledge based expert advice to managers for decision-making. All these have increased importance of MIS for the success of an organization.

Management Information Systems can help a business in that they contain important information about a particular client or event that takes place in the organization or the environment surrounding it. MIS is not as important for smaller organizations as it is for the larger corporations. The smaller locally run businesses are run usually by owners who rarely need instant access of information that larger companies require. Large corporate with varied product lines definitely can't do without a computer based MIS in order to survive and keep pace with competitors.

Any MIS performs various roles in an organization:

Supports day-to-day business operations;
Supports managerial decision-making;
Supports strategic decision-making and competitive advantage;
Optimizing operational cost;
Provide timely and accurate information; and
Provide expert advice to the managers on selected domains.

For example, an organization may use MIS to keep track of inventory, evaluate sales trend of different products, keep information about client and employee, etc.

3.2 Application of Management Information Systems

Management Information Systems are used for:

Operational Control: Information for control of day-to-day business operations. Information required by operational managers to control their daily work. This includes information on current stock of items, employee attendance, employee performance sheet, etc. Such information is very much structured and computational in nature and is produced in fixed format. Example: In an inventory control system, report on minimum inventory levels for reordering of

inventory, sales performance figures by product line, sales person or sales region can be obtained with the help of MIS.

Management Control: Information for short term planning (few weeks and months). Information is rather un-structured or semi-structured such as cash flow statement, sales trend analysis, monthly and annual financial statements .This type of information is used by mid-level manager for planning and control of organizational sub-units. Example: Sales trend figure in different regions of the country for product. Managers can carryout what if analysis like effect of price on sales figure, effect of cut on advel1isement on sales.

Strategic Planning: Information for long-term planning, developing policies and long-term goals for the organization. Such information is ad-hoc and unstructured such as human resource forecast, market trend analysis, etc. This type of information is mostly useful for top management.

SELF ASSESSMENT EXERCISE

What are the uses of MIS?

4.0 CONCLUSION

In this unit, you would have been introduced to the Management Information System. We have equally considered the role and application of MIS in an organization.

5.0 SUMMARY

What you have learned in this unit concerns the role and application of Management Information System (MIS).

6.0 TUTOR-MARKED ASSIGNMENT

Any MIS performs various roles in an organization, list these roles?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com

UNIT 10 TYPES OF INFORMATION SYSTEMS

CONTENTS

1.0 INTRODUCTION

Transaction processing management information, decision support and expert systems are different types of information systems considered in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

know various types of information systems
distinguish between the information systems
know the usefulness of MIS to managers.

3.0 MAIN CONTENT

3.1 Types of Information Systems

Depending on the end use, the information systems may be classified broadly to operation information systems and management information

systems. Operation information system generally helps to support business operation, whereas management information system helps in managerial decision-making. Transaction processing systems may be classified into Operation information system. Decision support systems, MIS and expert systems may be classified into different forms of MIS for specific purposes. The information requirements of managers are directly related to the position of manager in hierarchy ladder as shown in Figure 13.1.
Keeping this in view, various types of information systems have evolved over time.

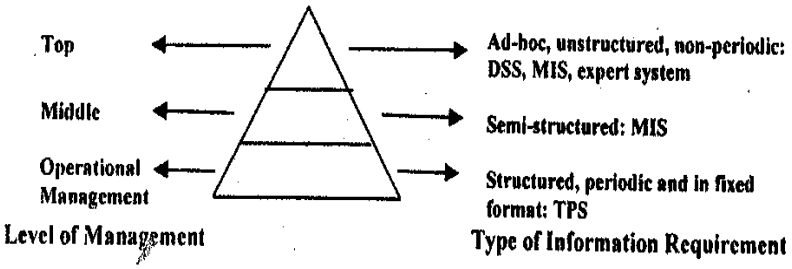


Figure 13.1: Levels of Management and their Information requirements

The types of information systems to be discussed subsequently have specific focus areas to support an organization's information requirements. Table 13.1 depicts various functions of Information Systems.

Table 13.1: Functions of Information Systems

Systems	Transaction Processing Systems	Management Information Systems	Decision Support System	Expert Systems
Information Source	Process Data resulting from business operation	Process data from business operation as well as external data	Use analytical models and specialized database in addition to internal data.	Use knowledge of experts from a specific field
Types of Support	Provides support for day-to-day operation of business process	Provides data for managerial decision-making	Provides interactive decision support to managers for decision-making	Provide expert advice on a specific domain of activity
Format of Reporting	Periodic and routine type in fixed format	Reports are semi-structured and ad-hoc type	Provides report like sensitivity analysis and what-if analysis	Provides advice like human expert
Used by	Operational management	Strategic decision-making for managers	For decision support tailored made to individual managers	Managers for expert advice on a specific field.
Examples	Sales transaction processing system, on-line railway reservation system	Marketing management information system	Geographic Information System (e.g. IBM's Geo-Manager, which integrates interacticve computer graphics with geographic database.	Expert System for medical diagnostic (e.g., MYCIN)

3.2 Transaction Processing System

Businesses offer service and products to the customers. In simple terms, transaction processing system is an information system that supports business in the delivery of various business transactions. A transaction processing system records and processes data resulting from business transaction. Transactions are events that occur as a result of business operations like transfer of money from one account to another account, purchase of items, etc. Transactions are basically a series of related operations that must all succeed or fail as a group. A single transaction of withdrawing money from a bank account actually involves two operations are a debit to an account and credit to another account. Transactions processing system allows the two operations to group into a single transaction. When both the operations are successfully completed, then the transaction is said to be complete. TPS can be classified into the category of Operation information system. Example can be Sales Transaction Processing System. These systems are transaction intensive and results of such transaction processing are used-to update various databases like customer databases, inventory databases

and accounts receivable databases. Transaction Processing Systems are also used to make day to day decisions that control operational processes.

Transaction processing systems (TPS) could be on-line or off-line. In case of On-line
Transaction Processing systems, data is processed by the system immediately after the transaction occurs. Point of Sale (POS) is a common example of the On-line Transaction Processing System (OL TP).

The following are major characteristics of Transaction Processing Systems:

Support business operations;
Focus on data resulting from business transactions; and
Captures and processes data of business transactions.

For example, consider a Sales Transaction Processing System. Sales Transaction Processing Systems handle routine business operations like sales and maintain records related to those activities. TPS transforms large number of inputs to outputs, using simple processing logic and operations. Compared to other types of information systems, TPS handles far more volume of inputs and outputs but generally involves simple processing logic.

Transaction Processing Systems perform the following operations (Refer to Figure 13.2):

Data is captured from documents or business operation and input into the system to record a transaction.
Then, data is processed. That is, calculations or other logical operations are performed for output.
The relevant files or databases are then updated with the results. Output of a TPS includes documents and reports.

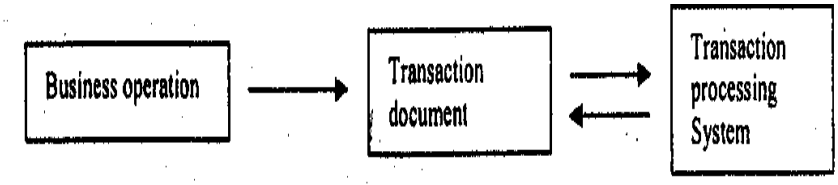


Figure 13.2: Business Operation and Transaction Processing System

To save time, storage space, and reduce errors of data entry, it is desirable to capture the information electronically at its point of origin, i.e. from the point of sales terminal (POS). This is referred to as source data automation. Rarely, non-conventional methods are used to facilitate data entry. For example, in a library, the barcode printed on the library member's card can be used to capture required information such as name of the member, address, validity date of the membership etc. Figure 13.3 depicts a Transaction Processing System at a Library.

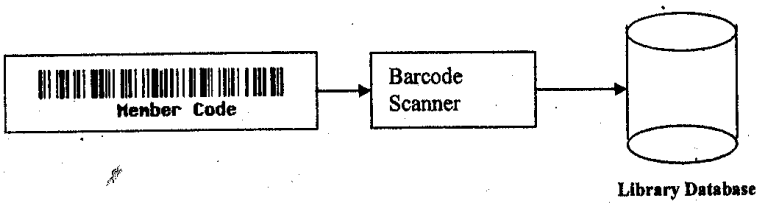


Figure 13.3: Transaction Processing System at a Library

The TPS should have the ability to process work flows of a business and each state of the business transaction can be represented by a step in the work flow. TPS captures and processes data of every business transaction and updates the relevant files and databases. It produces a variety of information for internal and external use.

Components of a Transaction Processing System

Consider a typical Transaction Processing System as depicted in Figure 13.4

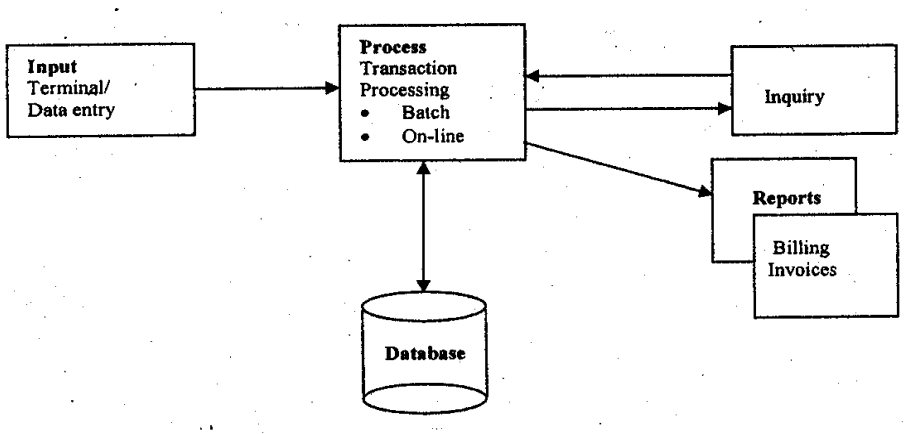


Figure 13.4: A typical transaction processing system

Data Entry: Data can be captured directly from machines which consist of data when it is entered during business transaction or the data can be directly keyed in. Sometimes, Data is also converted to a machine-readable form by scanning.

Transaction Processing: Input data is processed basically in two ways, namely Batch Processing or Online Processing. Table 13.2 compares the both.

Batch Processing: In this technique, accumulated data over a period of time is processed periodically depending on the requirement. This type of processing is economical. It is suited for situations when it is not required to process the transaction data as occurs and report generated are required only at a scheduled interval.

On-Line Transaction Processing (OLTP): In this technique, the data is processed by the system immediately after the transaction is over. This type of processing is well suited for small transactions and where turn around time is important. The database is always up-to-date since these are updated as when the transaction data is generated. Responses to the user inquiry are immediate. Since the database is accessed and updated after every transaction, care must be taken to protect the integrity of the database. Controls are some times built-in to the software for this kind of applications. Information systems related to Banking are examples of OLTP. The drawback of OLTP is high costs associated with the necessary security and fault tolerance features.

Table 13.2: Batch and Online Transaction Processing

	Batch Processing	Online Transaction Processing
Process	Transaction data is accumulated in regular intervals for processing at a scheduled interval	Transaction data is processed as and when generated by the business process
Updation of database	When the batch is processed	When the transaction is processed
Response time	Several hours/day	Immediate
Associated cost	Economical with efficient utilization of resources	High
Example	Processing pay cheques received for clearance in a banking system	Point of sales terminal, Online Railway reservation system

Database: It is the most important component of TPS. TPS updates the organizational database that reflects day-to-day business transactions carried out by the organization. For example, stock from the inventory database is reduced when an item is sold from POS stock. Stock from inventory database is increased when an item is

received. Debtor's and creditor's database is updated depending on amount received or paid in an accounts receivable database. The data generated and processed by TPS are subsequently used for other Information Systems such as Decision Support Systems etc.

Inquiry Processing: The transaction processing system supports query by the end-users. This inquiry processing is done by separate sub-system of the TPS. The user can make specific query by using the sending query to the inquiry processing system sitting on a LAN and receive response immediately.

Document and Report Generation: The final stage of the transaction processing system is document generation. The collection of documents generated by the TPS is called transaction document. Invoice generated by a POS terminal is an, example. Transaction logs are specific types of documents generated for Audit and other control purposes. All transactions recorded on the databases are printed.

Examples of transaction processing systems are sales transaction processing system (Refer to Figure 13.5), marketing transaction processing System, financial accounting system. One of the special types of transaction processing system is process control system (PCS). These are the systems that control processes of a manufacturing unit in a plant. Many processes have been mechanized by PCS minimizing human involvement.

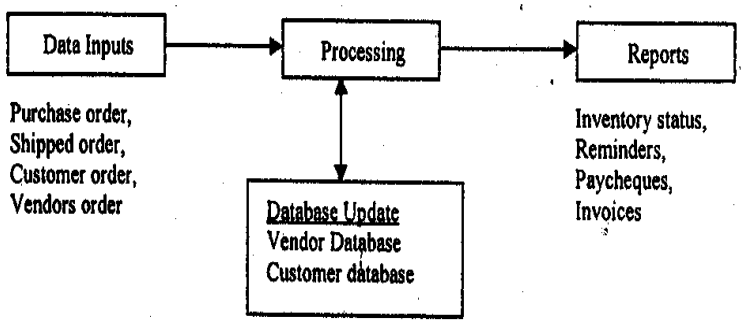


Figure 13.5: Sales transaction processing system

3.3 Management Information System

Management Information System (MIS) is a special kind of information system that helps managers to take decisions. MIS is tailored to provide specific information to individual managers for long term and strategic decision-making. MIS is used by the middle and top-management for their information requirements for decision-making. The use of MIS helps to produce the information that organizations need to improve decision-making, problem solving, controlling operations, and creating new products or services. Keeping this in view, a number of

organizations invest in development of a computerized MIS. The focus of MIS is to provide strategic information required by top-management. Major volume of information for top management comes from events not directly related to day-to-day business operations. Therefore, the information from normal reporting systems is found inadequate for managers. Therefore, special information system is developed for top management to support their activity, which is not met by other information systems.

The following are the characteristics of Management Information Systems (MIS):

Provides reports to management usually in semi-structured format (in detailed, summary, and exception);
Usually uses shared database from many sources;
Often based on management or statistical models;
Information presented in both textual and graphical forms, but more often in graphical format;
Provides information on trend analysis, exception reporting and what-if-analysis. It allows the user to ask questions such as, what if we increase price by 10%, the effect on sales of the product? If inflation increases by 5 Per cent what will be the effect on the sales forecast?

Table 13.3 depicts a Sales Performance Report. Figure 13.6 depicts a Bar chart only Figure 13.7 depicts a Pi chart.

Table 13.3: Sales Performance Report

Sales Region	2000	2000	2001	2001	2002	2002
	Estimated	Actual	Estimated	Actual	Estimated	Actual
East	54353	98877	435	76	76667	76776
West	5453	34534	43	59	867	64465
North	9876	5354	435	567	76667	76776
South	89987	98877	675	345	878	876

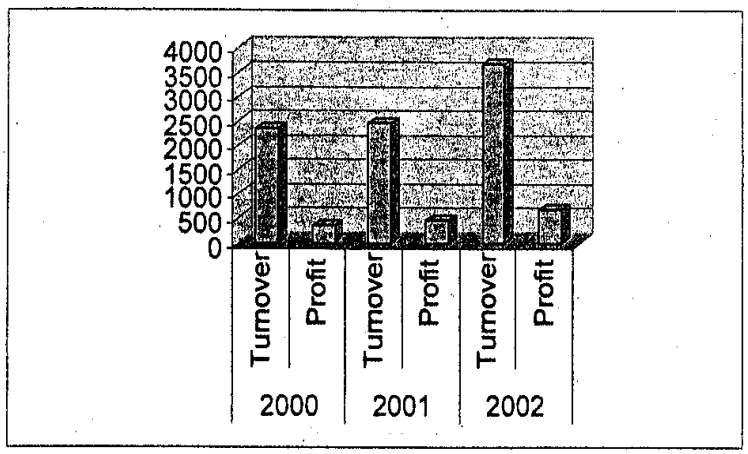


Figure 13.6: Turnover and Profit Ratio (Bar chart)

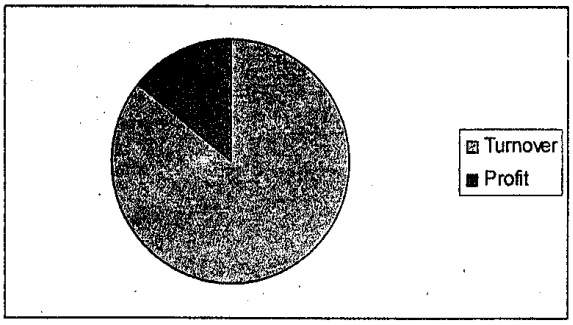


Figure 13.7: Turnover and Profit Ratio (pi chart)

Components of MIS

The bulk of information requirement of Managers at middle and top levels comes from external non-computer sources like meeting documents, newspaper, telephonic talk, letters, memos, etc. Corporate databases are important for day to day operations of the organization.

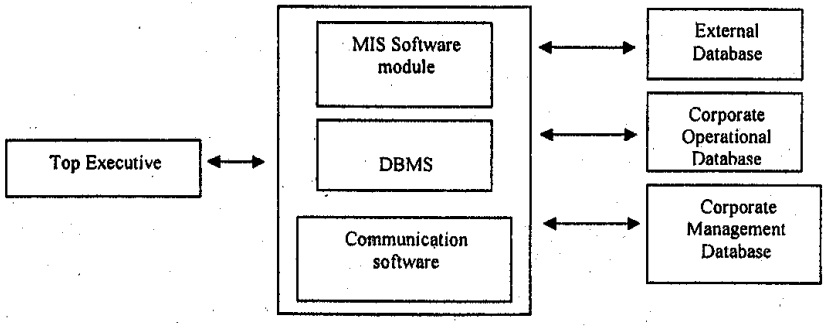


Figure 13.8: Components of an MIS

At the same time, data from external non computer sources provides managers with objective information that helps them to make strategic,

long and near term decisions. Various components of MIS are showed in Figure 13.8 and explained below.

External Database: External databases are databases that are not owned by the organization and the organization pays royalty to access these databases. Examples of these databases are: databases of Market research groups, Statistical and Demographic organizations etc. Since organization operates in a social environment it is influenced by various external factors. Impact of theses external factors on the long-term goal and success of organization is very important. Top management needs to analyse data from these sources for long term planning.

Corporate Database: Corporate database stores data generated by various business processes through transaction processing Systems. These can be employee database, customer database, inventory database, etc.

Management Database: These databases store select data from corporate databases. It generally stores summarized information for the requirements of managers.

MIS Software: This is used to extract and process information from various databases. It acts as a user interface to the managers.

DBMS: Database Management System stores, retrieves and manages data on various databases.
Communication Software: This is used to communicate with customers, suppliers and other stakeholders of the organization. Examples are Messaging Software or Organization's Bulletin board.

3.4 Decision Support System

In contrast to other information systems which provide general information about organization's performance in fixed format to managers, Decision Support Systems provide information to managers which will be helpful for them to make decisions. A manager at a higher level needs adhoc information for strategic planning and control.

Decision Support Systems (DSS) can be defined as a specific class of information systems that support business and organizational decision-making activities, as required by managers. A properly designed DSS is an interactive software-based system intended to help decision makers compile useful information from raw data, documents, personal knowledge, and/or business models to identify and solve problems and make decisions.

The following are the major characteristics of Decision Support Systems (DSS):

Help decision makers to take decisions rather than replace them;
Use underlying data and models;
Have little or no reasoning capability;
Are tailored to directly support decision-making styles of individual managers; Support interactive inquires and responses;
Are used to aid semi-structured or unstructured decisions; Produce information on ad-hoc, flexible and adaptable format; Information is produced by analysis of operational and external data; and
Analyses and supports comparison of specific alternative decisions.

Components of a DSS

Figure 13.9 depicts various components of a Decision Support System. They are explained below:

Data Management System

This is a system where various activities associated with retrieval, storage, and organization of the relevant data for the particular decision context is managed. It also provides security functions, data integrity procedures, backup and recovery, concurrency control, and general data administration. It can be a relational, objected oriented or any other suitable database.

Model Management System

Similar to Data Management Systems, Model Management Systems perform retrieval, storage, and organization activities associated with various quantitative models that provide the analytical capabilities for Decision Support Systems. This software module is responsible for analytical and limited reasoning capability of a DSS. This may contain various statistical and operation research models.

Knowledge Engine

This module is responsible for activities related to problem recognition and generation of interim or final solutions. The knowledge engine is the “brain” of the Decision Support System. Decisions require reasoning, and less structured decisions require more reasoning.

User Interface

This software module provides functionalities for input/output, error capturing and reporting. A common user interface for various Decision Support Systems is not possible as their designs vary in accordance with the environment of the organization when they are deployed.

Types of User interface: Keyboard, Joystick, Mouse, Scanner, Voice, Pen mouse, Touch screen, etc.

Like all information systems, issues related to the user such as training, skill, motivation levels are critical.

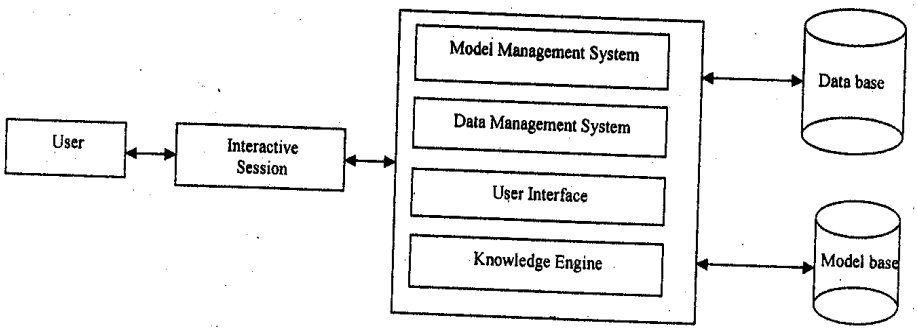


Figure 13.9: Components of Decision Support System

Types of DSS

Various Decision Support Systems are Communication driven DSS, Data driven DSS, Model driven DSS and Knowledge driven DSS. Table 3.4 draws a comparison between MIS and DSS.

Table 13.4: Management Information Systems and Decision Support Systems

	Management Information Systems (MIS)	**Decision Support Systems (DSS)**
Structure of Information	Periodic and often in fixed format	Interactive inquiry and Response to support
Source of information	Operational data/external database	Analytical models/external database and operational database
Target	Support group decision-making by managers	Tailored to decision-making style of individual managers

3.5 Expert Systems

An Expert System is a computer program that simulates the judgment and behaviour of a human expert or an organization that has expert knowledge and experience in a particular field, Typically, such a system contains a knowledge base containing accumulated experience and a set of rules for applying the knowledge base to each particular situation that is described to the program. Sophisticated expert systems can be enhanced with additions to the knowledge base or to the set of rules. The expert system is a knowledge-based information system to act as a consultant to the user. Expert systems are being used in many specialized field like medicine, engineering and business. An Expert System in the field of medicine can help diagnose illness. Unlike Decision Support System, an expert System interacts with the user to get input and provides expert advice on a problem in a specific domain.

Among the best-known expert systems have been those that play chess and those which assist in medical' diagnosis such as Mycin.

The following are the major characteristics of expert systems (ES):

Captures knowledge and expertise of a problem solver or decision maker and simulates thinking for those with less knowledge;
Replaces a human advisor/expert for specific domain of knowledge;
Its domain of knowledge is narrow;
Has reasoning and explanation capability;
Types of problem treated is repetitive; and
The direction of interaction is from machine to the user.

Expert systems are distinct from traditional Information Systems because of two main reasons:

Representation of Knowledge: Information is expressed in declarative form in contrast to procedural expressions used in other types of Information Systems. Here, knowledge is stored in a structured non-procedural way.

Perform Inexact Reasoning: Reasoning -A process by which new information is derived from a combination or combinations of existing, or previously derived, information. In this aspect, an expert system comes closer to human mind, which is hardly seen by traditional software. The ability to perform in exact reasoning leads to easier decision-making because irrelevant alternatives are reasoned out before the execution of the software.

Components of Expert System

An Expert System consists of a knowledge base and a software module (called inference engine) to perform inferences from the knowledge base. These inferences are communicated to the user. Figure 13.10 shows the components of an expert system.

Knowledge Base: It contains facts on a specific subject domain and rules to express the reasoning capability of a subject expert. Knowledge Base is logically divided into a fact base and a rule base. Knowledge means rules, heuristics (non-algorithmic), boundaries, constraints, previous outcomes and other knowledge programmed in by designers. A knowledge base typically incorporates definitions of factual knowledge and rules along with control information. Knowledge base format is specific to the implementation of the expert system software. Figure 13.11 shows the components of knowledge base.

Knowledge base contains much of the problem solving knowledge. Rules are of the form IF <condition> THEN <action>. Rules can be chained together (e.g., "If A then B" "If B then C" since A ⟶ B ⟶ C so "If A then C"). (If it is raining, then roads are wet. If roads are wet, then roads are slick.).

Inference Engine: Inference engine is software that provides the reasoning capability to the expert system. It processes rules and facts to provide advice on a specific problem. Rule based expert systems make use of two types of inferences for reasoning by forward chaining and backward chaining. Some expert systems use forward chaining by applying rules and facts to reach the conclusion where as others use backward chaining methods where it is verified whether the stated conclusion can be reached by applying the rules to the facts. The types of data processed by the inference engine are symbolic rather than numeric or character data types processed by other types of information systems. It usually takes the help of heuristic to solve a problem which other wise leads to combinatorial explosion.

In addition to above, expert systems may contain a knowledge acquisition module which in reality does not form the component of an expert system but is certainly important for development of knowledge base of an expert system. Special1y designed languages such as LISP and PROLOG are used for programming expert systems.

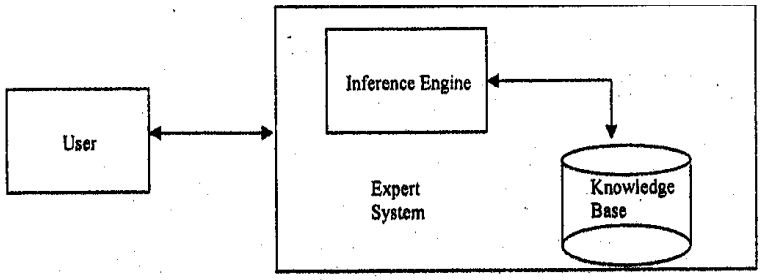


Figure 13.10: Components of an Expert System

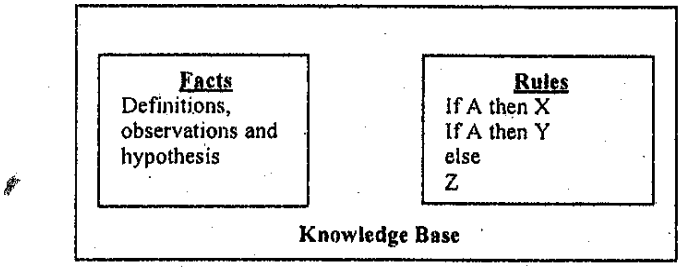


Figure 13.11: Components of Knowledge Base

An expert system starts with an interactive query session, which is directed from the expert system to the user. In this interactive query session, expert system asks a series of queries to the user and expects reply from the user sil1lilar to a doctor asking a series of queries to the patients before reaching any conclusion on the diagnosis of the disease. The user is expected to give reply to all the queries based on which the expert system recommends a solution like human expert. The advantage of computer based expert system is that it is unlike a human expert who is prone to environmental condition, these systems are consistent, fast and accurate in providing expert advice. It can also be programmed to give advice on behalf of several experts. This is the reason why expert systems are used as knowledge based strategic information resources for the managers in an organization. Various information systems are developed with an expert system component built in to it. These are called expert assisted information systems.

Knowledge Acquisition by Expert Systems

Expert systems must liase with people(experts) in order to gain knowledge and the people must be specialized in the appropriate area such as Medicine, Geology and Chemistry to name a few. Knowledge Engineer acts as an intermediary between the specialist (human expert) and the expert system. This process of picking the brain of an expert is a specialized form of data capture and makes use of interview techniques. The Knowledge Engineer is also responsible for the self-consistency of the data loaded to the expert system. Thus, a number of specific tests

have to be performed to ensure that the conclusions reached are sensible and accurate. Figure 13.12 depicts communication between expert system, knowledge base and human expert.

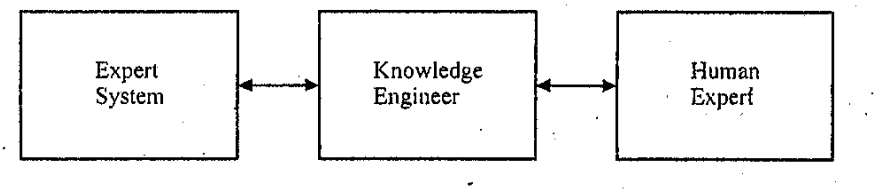


Figure 13.12: Communication between Expert System, Knowledge Engineer and Human Expert.

There are various applications for expert systems in business, engineering and medicine. Expert systems ask the user, a series of queries and based on the feedback from the user, deliver expert advice on the specific subject. Expert systems are used in the field of Medical diagnosis, Sales forecasting etc. Expert Systems are being used by managers for credit management, employees performance evaluation, portfolio analysis and production monitoring. Although expert systems are used in many fields, it can never replace a human expert. Expert system can provide expert advice based on the available information and knowledge. Expert systems lack learning capability like human being and have very limited focus area. It fails in the areas where advice requires a broad knowledge base.

Table 13.5 draws a comparison between Decision Support System and expert System.

Table 13.5: Decision Support Systems and expert Systems

	Decision Support System (DSS)	Expert System (ES)
Objective	To assist human decision	To mimic human decision
Reasoning capability	No or limited	Yes
Database	Adhoc, factual information	Procedural and factual knowledge
Domain	Broad	Narrow, very specific domain
Types of Data	Numerical, character based	Mostly symbolic
Direction of Query	Human to machine	Machine to human
Decision Maker	Human takes the decision with support from DSS	Computer makes the decision

SELF ASSESSMENT EXERCISE

Explain the application of management information system?

4.0 CONCLUSION

In this unit, you would have learned about the different types of information system, difference between the information systems and the usefulness of MIS to managers.

5.0 SUMMARY

What you have learned in this unit focuses on the different types of information systems.

6.0 TUTOR-MARKED ASSIGNMENT

Stated the functions of the Information Systems?

7.0 REFERENCES/FURTHER READINGS

Jeffrey L. Whilten, Lonmie D. Benltey, Kevin C, Diltoman; (2001). *System Analysis and Design Methods*, (Fifth Edition).

Jeffrey A. Hofter, Jorcy F. George, Joseph S. Valaclch, (2002). *Modern Systems Analysis and Design Pearson Education*; (Third Edition).

Elias M. Award; (1994.). *Systems Analysis and Design*; Galgotia Publications; (Second Edition).

Perry Edwards; (1993). *Systems Analysis and Design*; McGraw-Hill Publication.

Online Resources

http://www.rspa.com