

---

# UNIT 1 INTRODUCTION TO SAD

---

Structure	Page No.
1.0 Introduction	5
1.1 Objectives	5
1.2 Fundamentals of Systems	5
1.2.1 Important Terms related to Systems	
1.2.2 Classification of Systems	
1.2.3 Real Life Business Subsystems	
1.3 Real Time Systems	7
1.4 Distributed Systems	8
1.5 Development of a Successful System	9
1.6 Various Approaches for Development of Information Systems	11
1.6.1 Structured Analysis and Design Approach	
1.6.1 Prototype	
1.6.2 Joint Application Development	
1.7 Summary	15
1.8 Solutions/ Answers	15
1.9 Further Readings	16

---

## 1.0 INTRODUCTION

---

In general, a System is based on Input-Process-Output (IPO model). Manual work can be replaced by computerized system for accuracy and speed of processing. So, before the development of any computerized system, developers should also understand all basic concepts about the system. To develop a system, a standard Methodology must be considered. Different approaches are available for the development of a system. Selecting the best approach is the responsibility of systems analyst and this selection is based on the requirements of end user, problem definition and the infrastructure provided. Standard principles should be followed for the development of good quality software.

---

## 1.1 OBJECTIVES

---

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

- learn the concepts related to Systems;
  - know about Real Time Systems;
  - know about Distributed Systems; and
  - learn the process of developing a successful system.
- 

## 1.2 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.

### **1.2.1 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.

### **1.2.2 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(refer to Fig 1.1 in unit-1-pg-1.jpg)

- 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.

### 1.2.3 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 firm and its environment. In this system, all the subsystems work together to achieve a goal.

---

## 1.3 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.

## **Check Your Progress 1**

Select the appropriate choice given under each question.

1. \_\_\_\_\_ are comprehensive, multiple-step approaches to system development that will guide your work and influence the quality of your final product.
  - a) Techniques
  - b) Tools
  - c) Methodologies
  - d) Data flows.
2. The person in an organization who has the primary responsibility for systems analysis and design is \_\_\_\_\_.
  - a) The end user
  - b) The systems analyst
  - c) The internal auditor
  - d) Business manager.
3. An overall strategy to information systems development that focuses on the ideal organization data, rather than where and how data are used best defines the \_\_\_\_\_.
  - a) Process-oriented approach
  - b) Data-organization approach
  - c) Data-oriented approach
  - d) Information-oriented approach.
4. Which of the following is not a true statement concerning the differences between the process-oriented and data-oriented approaches to systems development?
  - a) The process oriented approach has limited design stability
  - b) Much uncontrolled data duplication exists with the data oriented approach
  - c) The data oriented approach designs data files for the enterprise
  - d) None of the above.
5. Non-information system professionals in an organization who specify the business requirements and who use software applications are called:
  - a) Programmers
  - b) Network managers
  - c) Code designers
  - d) End users.
6. Which of the following is not one of the four classes of information systems?
  - a) Transaction processing systems
  - b) Decision support systems
  - c) Expert systems
  - d) Production systems.

---

## **1.4 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 a distributed systems paradigm 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 Logic 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.

## Check Your Progress 2

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?

.....

.....

.....

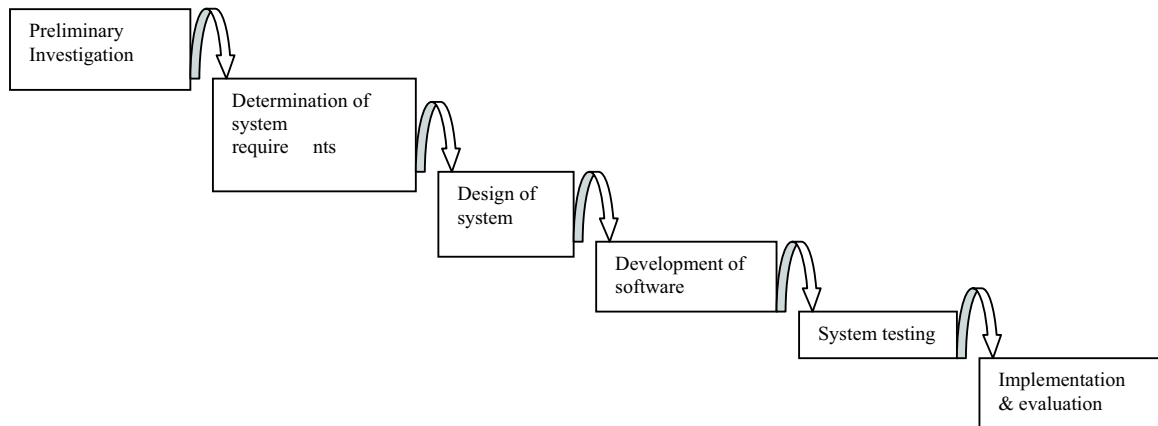
---

## 1.5 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 of a 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.
- (6) *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).
- (7) 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 (refer to Fig 1.4 in unit-1-pg2.jpg)

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.

---

## 1.6 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.

### **1.6.1 Structured Analysis and Design Approach**

The goal of structured system analysis and design is to 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 fulfil 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 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.

## 1.6.2 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 original 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).

### 1.6.3 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) **Managers:** 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.

### Check Your Progress 3

1. List the fundamental principles of S/W Development Life Cycle.  
.....  
.....  
.....  
.....  
.....

---

## 1.7 SUMMARY

---

In this unit, all the basic concepts that are necessary to understand the system, types and characteristics are given. Concepts about real and distributed systems are also discussed. Development of a successful information system depends on principles of SDLC. Different approaches are available for development of information systems. These are Model Driven (structure analysis and design approach), Accelerated (prototype) and JAD approach. Selection of the approach is based on the end user requirements, problem identified and infrastructure provided.

---

## 1.8 SOLUTIONS/ ANSWERS

---

### Check Your Progress 1

1. c
2. b
3. c
4. b
5. d
6. d

### Check Your Progress 2

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

**Strategic Information:** This is the information needed for long range planning and top-level management of any organization that uses it. This information is unstructured or semi-structured and the volume of this information in any organization is very small.

**Tactical Information:** This type of information is needed to take short term decisions to run the business efficiently and middle level managers' use it. The volume of tactical information is more than the volume of strategic information.

**Operational Information:** This type of information is used for day-to-day operations of a business organization and first level management in the organization uses it. It is usually easy to obtain by straightforward clerical processing of data. The volume of this information is much more than volume of tactical information.

**Repository:** It is defined as data store of accumulated system knowledge i.e. system models, detailed specifications, and any other documentation that has been accumulated during the system's development. This knowledge is reusable and critical to the production system's ongoing support. The repository is implemented with various automated tools and it is often centralized as an enterprise business and IT resource.

2. Information requirement determination attempts to find the strategic, tactical and operational information that is needed to effectively manage an organization. Information specification defines the manner in which the information will be presented and the analyzed data, it consists of.
3. The following are characteristics of a good information system:
  - meeting customer's requirements,
  - modular design, and
  - easily maintainable.

### **Check Your Progress 3**

1. The following are the fundamental principles of SDLC:
  - Management and users should be involved because they can explain the problem that is to be taken for design and development in accurate manner.
  - A problem solving approach should be adopted
  - Phases and activities should be established.
  - Some standards should be established for consistent development of System.
  - Development of Information System should be considered as capital Investment.
  - Divide and Conquer approach should be adopted. It is one way of making complex problem easier.
  - For making a system successful, it should be designed for growth and change.

---

## **1.9 FURTHER READINGS**

---

- Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *System analysis and design methods*; Tata McGraw-Hill ;Fifth Edition;2001.
- By Jeffrey A. Hoffer , Joey F. George , Joseph S. Valacich; *Modern Systems Analysis and Design*; Pearson Education; Third Edition;2002.

#### **Reference Web sites**

<http://www.rspa.com>

---

## UNIT 2 SYSTEMS ANALYST – A PROFESSION

---

Structure	Page No.
2.0 Introduction	17
2.1 Objectives	17
2.2 Why Do Businesses Need Systems Analysts?	18
2.3 Users	18
2.4 Analysts in various functional areas	19
2.4.1 Systems Analyst in Traditional Business	
2.4.2 Systems Analyst in Modern Business	
2.5 Role of a Systems Analyst	20
2.6 Duties of a Systems Analyst	21
2.7 Qualifications of a Systems Analyst	22
2.7.1 Analytical Skills	
2.7.2 Technical Skills	
2.7.3 Management Skills	
2.7.4 Interpersonal Skills	
2.8 Summary	28
2.9 Solutions/ Answers	28
2.10 Further Readings	29

---

### 2.0 INTRODUCTION

---

The work of a systems analyst who designs an information system is the same as an architect of a house. Three groups of people are involved in developing information systems for organizations. They are managers, users of the systems and computer programmers who implement systems. The systems analyst coordinates the efforts of all these groups to effectively develop and operate computer based information systems.

Systems analysts develop information systems. For this task, they must know about concepts of systems. They must be involved in all the phases of system development life cycle i.e. from preliminary investigation to implementation. Success of development depends on skills and the dedication of Systems analysts.

Analysing, designing and implementing systems to suit organizational needs are the functions of systems analyst. S/he plays a major role in evaluating business benefits from computer technology. Systems analyst is basically a problem solver with unique skills. A systems analyst deals with people, procedures and technologies.

---

### 2.1 OBJECTIVES

---

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

- know the need of systems analyst in the business and in the development of information system;
- know the job responsibilities of systems analysts in the traditional business and in the modern business;
- know the role of systems analyst in a team for the benefit of the organization and success of developed information system;
- know 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;

- know the way systems analysts use their skills in managing resources, projects, risk, and change; and
- know the importance of interpersonal skills for a systems analyst, in communicating, working with teams, facilitating groups, and managing exceptions.

---

## 2.2 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.

---

## 2.3 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, 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 connect their purchasing systems directly to the order processing systems of their suppliers then both become external users to each other.

---

## 2.4 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.

### 2.4.1 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** : In traditional 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.

### 2.4.2 Systems Analyst in Modern Business

Many medium-to-large information services units for the modern business have reorganized to be decentralized with a focus on empowerment and dynamic teams. In modern 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 modern information services organization.

In modern 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.

### Check Your Progress 1

1. What are the differences between problem identification and problem solving?  
.....  
.....  
.....
2. What is the difference between a logical system description and a physical system description?  
.....  
.....  
.....
3. Why is the development of information systems, sometimes done by an independent consultant?  
.....  
.....  
.....
4. Differentiate between systems analyst and a business analyst.  
.....  
.....  
.....

---

## 2.5 ROLE OF A SYSTEMS ANALYST

---

The success of an information 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 information 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 a 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.

---

## 2.6 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 analyst 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.

### Check Your Progress 2

1. Are excellent programmers necessarily excellent systems analysts? Justify your answer.  
.....  
.....  
.....
2. List at least eight tasks performed by systems analysts.  
.....  
.....  
.....
3. List at least six attributes of a systems analyst.  
.....  
.....  
.....  
.....
4. Why should a systems analyst be able to communicate well?  
.....  
.....  
.....  
.....

---

## 2.7 QUALIFICATIONS OF A SYSTEMS ANALYST

---

A systems analyst must fulfil 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.

### **2.7.1 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), s/he 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) How work officially gets done in a particular organization: In this area, knowledge about the following is required:

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

- (2) Understanding the organization's internal politics: 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 missions.

**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.

## **2.7.2 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.

### 2.7.3 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.

## **2.7.4 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 analyst 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 (refer to Fig 2.3 in unit-2.jpg)

### Check Your Progress 3

1. What are the business and technology trends that affect the players in the field of information systems?

.....

.....

.....

---

## 2.8 SUMMARY

---

In this unit, we discussed the skills necessary for success as a systems analyst. An organization needs systems analyst for the replacement of existing system with computerized system. A systems analyst may work on a project basis or may be part of client's team as a permanent employee who works about changes to be implemented to the existing system in the client organization. A systems analyst takes various roles to work in a team for the benefit of the organization and to develop successful information systems. Some of the roles are: Change Agent, Investigator and Monitor, Architect, Psychologist, Motivator, Intermediary.

The requisite skills for systems analyst are analytical, technical, management and interpersonal.

---

## 2.9 SOLUTIONS/ ANSWERS

---

### Check Your Progress 1

1. In problem identification, a systems analyst compares the current situation in an organization to the desired situation. Problem identification involves measurement, not decision making. Problem solving is the process of finding one or more ways to reduce these differences and then select the best approach for implementation.
2. A *logical system description* portrays the purpose and function of the system without tying the description to any specific physical implementation. A *physical system description* of a system focuses on how the system will be materially constructed.
3. Time constraints, limited number of internal personnel and need for a better expertise are reasons for hiring an outside consultant.
4. A systems analyst facilitates most of the activities to develop or acquire an information system. S/he studies the problems and needs of an organization to determine how people, data, processes, communication and information technology can best accomplish improvement of the business.

A business analyst is a systems analyst who specializes in business problem analyses and technology independent requirements analysis.

### Check Your Progress 2

1. An excellent programmer is not necessarily an excellent systems analyst. A programmer is given clear specification (often in writing) and designs efficient and maintainable programs. S/he need not have good communication skills and inter-personal relations. S/he need not have knowledge of functionality of organizations.  
A programmer works with clear specifications whereas an analyst has to arrive at clear specifications from fuzzily stated requirements.
2. Tasks performed by systems analysts are:
  - Define requirements,
  - Draws up ordering of requirements,
  - Gather data, facts and opinion of users,
  - Analyses the existing systems in the organization and uses this knowledge to improve systems,



- Provides solutions to problems posed by management,
  - Makes specification of information systems,
  - Designs the information system, and
  - Evaluates the information system.
3. The desirable attributes of a systems analyst:
- Must know how organizations function,
  - Must know latest developments in computer hardware and software,
  - Can get along with diverse people from top level managers to the last level of employees,
  - Must be able to express himself and absorb information by being a good listener,
  - Must have an analytical mind, and
  - Must have a broad general knowledge.
4. S/he has to understand the users' requirements mostly by interviewing them and thus s/he has to ask the right questions, listen carefully and summarize the gist of conversation. S/he should also be able to present and explain orally to the users, the system designed by her/him and clarify doubts they may have after the oral presentation. His main job is to interact with the management, users' and the programmers. So, it is obvious that s/he must possess good communication skills.

### Check Your Progress 3

1. Players in the field of information systems are being affected by a number of business and technology trends including the following:
- Total quality management, a comprehensive approach to facilitating quality improvements and management within a business.
  - Business process redesign, the study, analysis, and redesign of fundamental business processes to reduce costs and/or improve value added to the business.
  - Continuous process improvement, the continuous monitoring of business processes to reduce costs and add value.
  - Enterprise resource planning, the selection and implementation of a single vendor's fully integrated information system that spans most basic business functions required by a major corporation.
  - Electronic commerce, a new way of doing business that involves the conduct of both internal and external business over the internet, intranets and extranets.

---

## 2.10 FURTHER READINGS

---

- Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *System analysis and design methods*; Tata McGraw-Hill; Fifth Edition; 2001.
- By Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich; *Modern Systems Analysis and Design*; Pearson Education; Third Edition; 2002.

### Reference Websites

- <http://www.freeforessays.com>
- <http://www.rspa.com>



---

## UNIT 3 PROCESS OF SYSTEM DEVELOPMENT

---

Structure	Page No.
3.0 Introduction	30
3.1 Objectives	30
3.2 Systems Development Life Cycle	31
3.3 Phases of SDLC	32
3.3.1 Project Identification and Selection	
3.3.2 Project Initiation and Planning	
3.3.3 Analysis	
3.3.4 Logical Design	
3.3.5 Physical Design	
3.3.6 Implementation	
3.3.7 Maintenance	
3.4 Products of SDLC Phases	35
3.5 Approaches to Development	36
3.5.1 Prototyping	
3.5.2 Joint Application Design	
3.5.3 Participatory Design	
3.6 Case Study	37
3.7 Summary	43
3.8 Solutions/ Answers	43
3.9 Further Readings	45

---

### 3.0 INTRODUCTION

---

Information Systems Analysis and Design is a complex and stimulating process that is used to develop and maintain computer based information systems. The analysis and design of information systems are driven from an organizational point of view. An organization might consist of whole enterprise, specific departments or individual work groups. Information Systems Analysis and Design is, therefore, an organizational improvement process. Systems are built and rebuilt (enhanced) for organizational benefits. Benefits result by adding value during the process of creating, producing and supporting the organization's services and products. Thus, Information Systems Analysis and Design is based on the understanding of objectives, structure and processes of organization and the knowledge about the application of Information Technology for this purpose. Most organizations find it beneficial to use standard sets of steps, called a systems development methodology, to develop and support their information systems (IS). Like many processes, the development of Information Systems often follows a life cycle called Systems Development Life Cycle. For example, a product follows a life cycle when it is created, tested and introduced in the market. Its sale increases and goes to peak point and after that it declines and a new product or next version of the existing product is introduced in the market to replace it. SDLC is a common methodology for systems development in many organizations, consisting of various phases that mark the progress of system analysis and design effort.

---

### 3.1 OBJECTIVES

---

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

- define Information Systems Analysis and Design;
- describe the information systems development life cycle; and

- list alternatives to SDLC and compare the advantages and disadvantages of SDLC to its alternatives.

## 3.2 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.

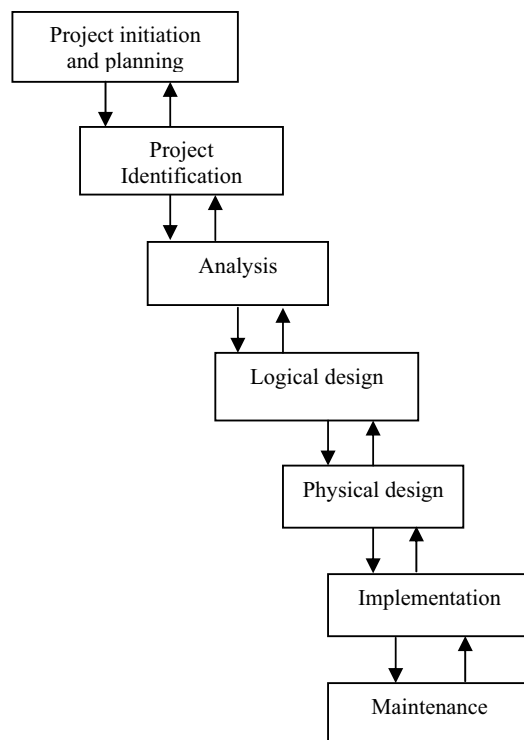


Figure: 3.1: Phases of System Development Life Cycle

Every custom software producer will have its own specific detailed life cycle 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. Each phase 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.

---

## **3.3 PHASES OF SDLC**

---

SDLC 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.

### **3.3.1 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 are 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 planning 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 terms of an initial study.

### **3.3.2 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 is 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 phases 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.

### 3.3.3 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 by the 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.

### 3.3.4 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.

### 3.3.5 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, and

processing of 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.

### **Check Your Progress 1**

1. What is the difference between Project Identification and Project Initiation?

.....

.....

.....

2. What is the difference between analysis and design?

.....

.....

.....

3. Why do most of the projects die after feasibility phase?

.....

.....

.....

### **3.3.6 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 tests 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 best 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.

### 3.3.7 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.

---

## 3.4 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.

---

## **3.5 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.

### **3.5.1 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 prototyping 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.

### **3.5.2 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 called 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.

### 3.5.3 Participatory Design (PD)

Participatory Design (PD) represents a useful alternative approach to the SDLC PD 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.

### Check Your Progress 2

1. Designing and building a scaled down but fundamental version of a desired system is the process known as \_\_\_\_\_
2. Prototyping is a form of \_\_\_\_\_
3. Implementation includes \_\_\_\_\_

---

## 3.6 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.

## Analysis

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.

## 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 various 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 –
	<ul style="list-style-type: none"> <li>• I =&gt; Book is issued.</li> <li>• L =&gt; Book is lost.</li> <li>• P =&gt; Book is present.</li> <li>• M =&gt; Book is lost by member.</li> <li>• D =&gt; Book is deleted.</li> </ul>

- **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 –
	<ul style="list-style-type: none"> <li>• I =&gt; book is issued for home.</li> <li>• R =&gt; book is issued for reading in library.</li> </ul>
State	State can be –
	<ul style="list-style-type: none"> <li>• 0 =&gt; book is with the member.</li> <li>• 1 =&gt; member returned the book.</li> <li>• 2 =&gt; member lost the book.</li> </ul>

- **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 of issue of the book to the member.
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 brought 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. <ul style="list-style-type: none"> <li>• P =&gt; Member can access the library.</li> <li>• D =&gt; Member is deleted for library access.</li> </ul>

- ***Member\_Deleted***

It stores information about the member 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.

The Zero level DFD is depicted in Figure 3.2.

Figure 3.2: 0- level DFD of Library Information System (refer to Fig. 3.2 in unit-3.jpg)

### 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.

### 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*** – (BookId, Name, Author, Category, Subject, Price, Date, Copies, Edition).
- ***Book\_Copy*** – (BookId, BookIDC).
- ***Book\_IR*** – (TID, BookId, MemId, Mode, State).
- ***Book\_TRDetail*** – (TID, Issue\_Date, Issue\_Time, Return\_Date, Return\_Time).
- ***Book\_Delete*** – (BookId, Date).
- ***Book\_Resume*** – (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\_State** – (MemID, State).
- **Member\_Delete** – (MemID, Date).
- **Member\_Resume** – (MemID, 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.

### Check Your Progress 3

1. What are the phases involved in the analysis and development of the system?  
.....  
.....  
.....
2. What are the basic types of feasibility studies used in system analysis and design?  
.....  
.....  
.....
3. "Implementation of system can be done in three different ways". Do you agree with this statement? Justify your answer.  
.....  
.....  
.....
4. What are various types of maintenance.  
.....  
.....  
.....

---

## 3.7 SUMMARY

---

In this unit, you learned about the basic framework that guides systems analysis and design, the systems development life cycle, with its seven major phases: project identification and selection, project initiation and planning, analysis, logical design, physical design, implementation, and maintenance. The life cycle has had its share of criticism, which you read about, and other frameworks have been developed to address the life cycle's problems. These alternative frameworks include: Prototyping (Rapid Application Development approach), Joint Application Design and Participatory Design.

---

## 3.8 SOLUTIONS/ ANSWERS

---

### Check Your Progress 1

1. The first phase in the SDLC is called project identification. In this phase, the user identifies the need for a new or improved system. Information requirements of the organization as a whole are examined, and projects to meet these requirements are proactively identified. In project initiation, the formal, but



preliminary investigation of the system problem or opportunity at hand and the presentation of reasons as of why the system should or should not be developed by the organization is done. A critical step at this point is determining the scope of the proposed system.

2. During analysis phase, the requirements are determined. In this phase, analysts work with users to determine what the users want from a proposed system. This phase usually involves a careful study of any existing systems, manual or computerized that might be replaced or enhanced as part of the project. Next, the requirements are studied and structured according to their inter-relationship and eliminate any redundancies. After this alternative initial design is generated to match the requirements then these alternatives are compared to determine which alternative best meets the requirements.

After analysis phase is complete, design of the system begins. The design consists of logical and physical design of the system. During design, the analysis converts the description of the recommended requirements into logical and then physical system specifications. Design occurs in two phases, viz., logical design and physical design.

Logical design concentrates on the business aspects of the system.

In physical design, the logical design is turned into physical or technical specifications.

3. If the proposed system is not feasible to develop, it is rejected at this very step. In this phase, feasibility study of the proposed system is performed. The Cost Benefit Analysis of the proposed system is prepared by the system analyst in this phase and if the cost of the proposed system in terms of time, money and resources outweigh the benefits of the system, then the proposed system is rejected.

### **Check Your Progress 2**

1. Prototyping
2. Rapid Application Design
3. Coding, testing and installation

### **Check Your Progress 3**

1. The following phases are involved in the analysis and development of the system:
  - Project identification and selection
  - Project initiation and planning
  - Analysis
  - Logical design
  - Physical design
  - Implementation
  - Maintenance.
2. The six basic types of feasibilities used in system analysis and design are:
  - Technical feasibility
  - Economic feasibility
  - Behavioural feasibility
  - Operational feasibility
  - Legal feasibility
  - Time feasibility.
3. Yes. The implementation of system can be done in three different ways. They are:

- Direct conversion
  - Parallel conversion
  - Phased conversion.
4. Types of maintenance are:
- Corrective maintenance
  - Adaptive maintenance
  - Perfective maintenance

---

## 3.9 FURTHER READINGS

---

- Kendall & Kendall; *Systems Analysis and Design*; PHI; Fifth Edition.
- Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *System analysis and design methods*; Tata McGraw-Hill; Fifth Edition; 2001.

### Reference Websites

- <http://www.rspa.com>
- <http://www.ieee.org>

---

## UNIT 4 INTRODUCTION TO DOCUMENTATION OF SYSTEMS

---

Structure	Page No.
4.0 Introduction	46
4.1 Objectives	46
4.2 Concepts and Process of Documentation	46
4.3 Types of Documentation	49
4.3.1 System Requirements Specification	
4.3.2 System Design Specification	
4.3.3 Test Design Document	
4.3.4 User Manual	
4.4 Different Standards for Documentation	58
4.5 Documentation and Quality of Software	60
4.6 Good Practices for Documentation	60
4.7 Summary	61
4.8 Solutions/ Answers	61
4.9 Further Readings	61

---

### 4.0 INTRODUCTION

---

Documentation is a process to help users of software and other people to use and interact with system. Effective and accurate documentation is very necessary for the success of any system. Documentation becomes part of each step of system development through out the process of system development even before the documentation starts officially. During the process of system development, study reports and other necessary information are documented to help people involved in system development to understand the process. There are many kinds of documentation namely analysis documentation, design documentation, interface documentation, internal program documentation and user-oriented documentation. Documentation should not be seen as a overhead for system development. Rather, documentation has to be seen as an investment for the development of high quality software.

---

### 4.1 OBJECTIVES

---

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

- understand the concept of documentation;
  - understand the importance of documentation;
  - learn about various documents and process of documentation;
  - understand application of various standards of documentation processes;
  - differentiate between various documentation processes;
  - know relation between the documentation and quality of software; and
  - understand the good practices for documentation process.
- 

### 4.2 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 12207:1995 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 audits and other requirements of the organization;
- to meet regulatory demand;
- needed for IT infrastructure management and maintenance; and
- needed for migration to a 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 documentation is 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.

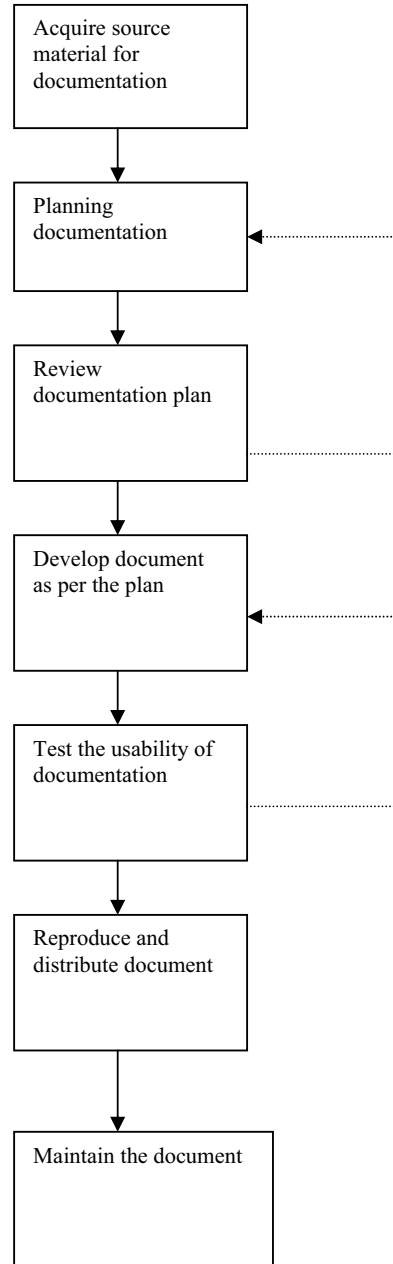
*Review of Plan:* The plan as set out in the process above is reviewed to see that material acquired is correct and complete.

*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**

### **Check Your Progress 1**

1. ISO standard ISO/IEC 12207:1995 describes documentation as a \_\_\_\_\_ activity.
2. The documenter is responsible for preparation of a documentation plan (Yes/No).

## 4.3 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.

### 4.3.1 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 addresses 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 constraints.
- 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.

## 7. Appendix

- Supplementary information to the specification.

### 4.3.2 System Design Specification

The system design specification or software design specification as referred to has 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 aspects 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 are used to explore the behaviours of one or more objects throughout a given period of time. This diagram is specifically useful to describe logic circuits.

### 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.

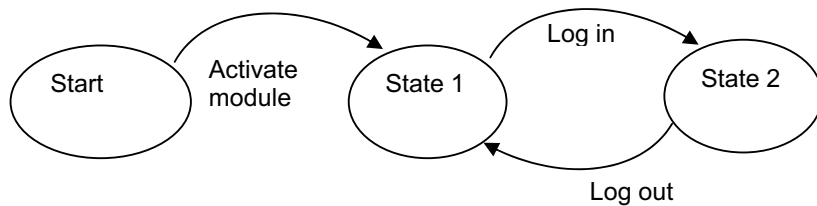


Figure 4.2 : A state machine diagram

### 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.

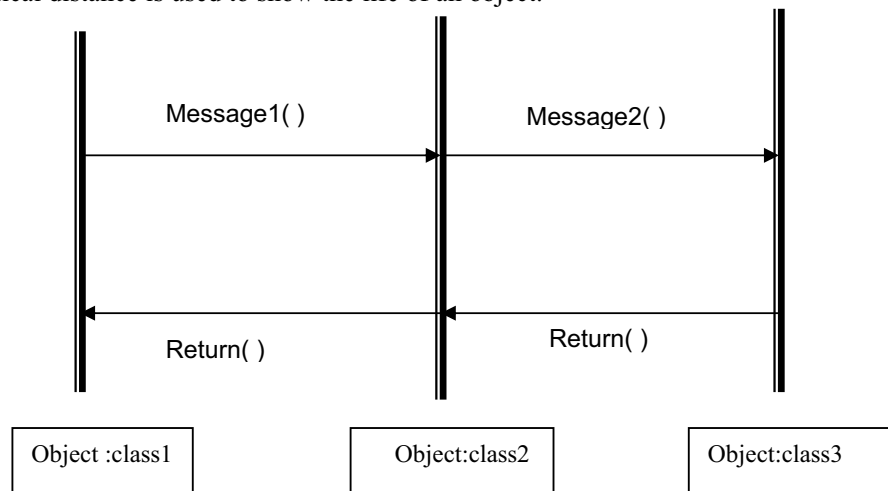


Figure 4.3: An object interaction diagram

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

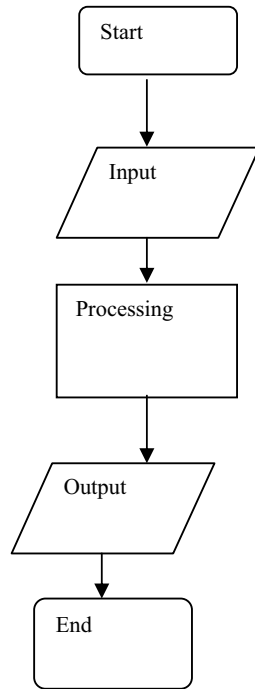


Figure 4.4 : Flow chart

### 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.

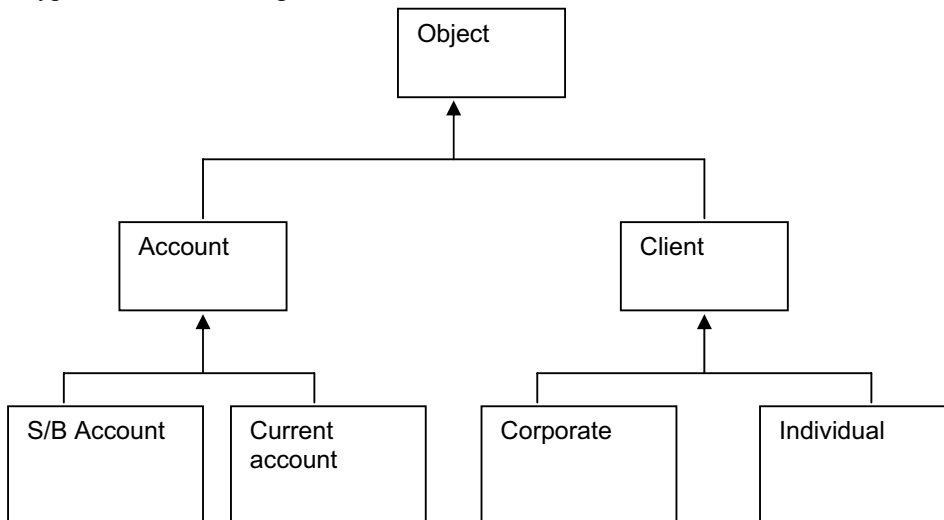
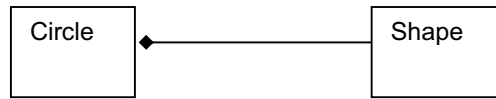


Figure 4.5: A typical inheritance diagram

### 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

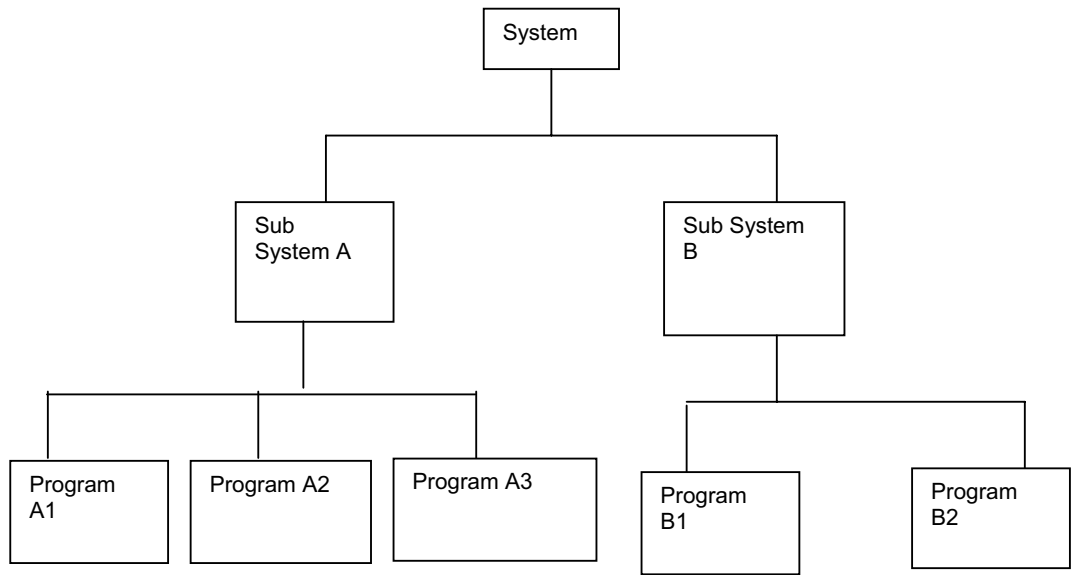
between these classes. Each module will be represented by its name. The relationship will be indicated by a directed line from container to container. The directed line is labelled with the cardinality of the relationship. It describes “has a” relationship. Figure 4.6 shows an aggregation between two 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 structures 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 HoursWorked > MaxWorkHour THEN
    Display overtime message
ELSE
    Display regular time message
ENDIF
```

**Figure 4.8 : Example of a Pseudocode**

### **Contents of a typical System Design Specification document content**

1. Introduction
  - 1.1 Purpose and scope of this document:  
Full description of the main objectives and scope of the SDS document is specified.
  - 1.2 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
  - 2.1 Overview of modules, components of the system and sub-systems
  - 2.2 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.

### **4.3.3 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 (R1993) 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.

### *2.1. Schedules and Resources*

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

### *2.2. 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.

### *2.3. 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**

### *3.1. Unit Testing*

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

### *3.2. Integration testing*

Integration test is done on modules or sub-systems.

## 4. Validation Testing

### 4.1. System Testing

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

### 4.2. 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.

## 4.3.4 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

#### 1.1 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.

#### 1.2 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.

#### 1.3 Glossary

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

#### 1.4 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.

#### 1.5 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.

### *2.1 System Usage*

Provide examples of normal usage. Images of the screendumps 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**

### *3.1 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.

### *3.2 Error Messages and Recovery*

Provides an *alphabetical* list of 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.

## **Check Your Progress 2**

1. The system design specification is developed in \_\_\_\_\_ process.
2. Pseudo code is used to describe \_\_\_\_\_.
3. \_\_\_\_\_ provides information as to how to operate and maintain the system.

---

## **4.4 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:

1. ISO/IEC 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.

2. ISO/IEC 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.

3. 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 development 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.

### **Check Your Progress 3**

1. \_\_\_\_\_ is used in the organization for uniform practices for documentation.
2. International Standard ISO/IEC 12207 is documentation standard (Yes/No).

---

## **4.5 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.

---

## **4.6 GOOD PRACTICES FOR DOCUMENTATION**

---

1. *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.
2. *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.

### Check Your Progress 4

1. SEI CMM is a \_\_\_\_\_ model.
  2. One of the basic goals of software engineering is \_\_\_\_\_.
- 

## 4.7 SUMMARY

---

Documentation is an integral part of software development process and should not be taken lightly. Incomplete and inaccurate documentation may pose serious hurdle to the success of a software project during development and implementation. The documentation process itself requires proper planning like the software development process. Various documents like System requirement specification (SRS), system design specification (SDS), test design document and user manuals are produced during the life cycle of a software development process. SRS documents the high level requirements of the system without going into details of implementation issues, whereas system design document describes how the requirements are finally to be implemented. It also describes the implementation issues with help of various system design tools. Test design document documents the requirement to be tested and procedure to be followed for testing. We have also discussed various ISO and IEEE standards on user documentation for uniform practices on documentation style and process. The relation of documentation with software quality has also been discussed.

---

## 4.8 SOLUTIONS/ ANSWERS

---

### Check Your Progress 1

1. Support
2. Yes

### Check Your Progress 2

1. Two stage.
2. Algorithms.
3. System administration guide.

### Check Your Progress 3

1. Software documentation standard.
2. No.

### Check Your Progress 4

1. Capability Maturity.
  2. To produce accurately working software along with the best possible supporting documentation.
- 

## 4.9 FURTHER READINGS

---

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

**Reference Websites**

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

---

## UNIT 5    PROCESS OF SYSTEMS PLANNING

---

Structure	Page Nos.
5.0 Introduction	5
5.1 Objectives	5
5.2 Fact Finding Techniques	5
5.2.1 Interviews	
5.2.2 Group Discussions	
5.2.3 Site Visits	
5.2.4 Presentations	
5.2.5 Questionnaires	
5.3 Issues involved in Feasibility Study	9
5.3.1 Technical Feasibility	
5.3.2 Operational Feasibility	
5.3.3 Economic Feasibility	
5.3.4 Legal Feasibility	
5.4 Cost Benefit Analysis	11
5.5 Preparing Schedule	12
5.6 Gathering Requirements of System	13
5.6.1 Joint Application Development	
5.6.2 Prototyping	
5.7 Summary	15
5.8 Solutions/Answers	16
5.9 Further Readings	16

---

### 5.0 INTRODUCTION

---

In this unit, you will learn the process of planning the development of systems. You will also learn various techniques used in it. Here, you will learn about the various techniques of fact finding and getting appropriate information about system which is currently in place. Based on the information gathered, a list of requirements has been compiled. The same is sent to the customer for his/her review and comments. The topic of feasibility study is discussed in depth in this unit. The process of cost benefit analysis is also discussed in this unit. Lastly, we shall discuss about the Joint Application Development (JAD).

---

### 5.1 OBJECTIVES

---

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

- know the various fact finding techniques and also their advantages and disadvantages;
- identify check points in system life cycle to conduct feasibility study;
- know the process of doing cost benefit analysis of the project;
- know various issues related with system development; and
- learn the process of preparing schedule.

---

### 5.2 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.

### **5.2.1 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 be 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 can be done at this point of time.

### **5.2.2 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 the problem at hand. Ultimately, they 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.

### **5.2.3 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.

### **5.2.4 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, the 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.

### **5.2.5 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.

The following are various types of Fixed format questions:

1. True / False or yes/no type questions.
2. Questions whose response will be one of the choices: strongly agree, agree, disagree..
3. Ranking type questions (ranking items in order of importance).
4. Multiple choice questions (select one response or all the relevant responses).

### Check Your Progress 1

1. .... and ..... are two types of interviews.
2. .... are special purpose documents that allow the analyst to collect information and opinion from respondents.
3. The engineers of the ..... organization make site visits.

---

## 5.3 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.



### 5.3.1 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, the manpower should be trained in the ERP software. This may lead to slippage in the delivery schedules.

### 5.3.2 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 the 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.

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.

**5.3.3 Economic Feasibility**

It is the measure of cost effectiveness of the 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.

**5.3.4 Legal Feasibility**

Legal feasibility studies issues arising out of the need to the 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 user's licences, single user licences, 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.

---

**5.4 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, the 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.1.5 lakhs per year. Interest of the investment is Rs.1,26,000 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 the project is feasible.

### Check Your Progress 2

1. .... consists of activities which determine the existence of scope of developing an information system to the organization.
2. .... is all about problems that may arise during operations.
3. .... and .... are two components of economic feasibility.

---

## 5.5 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.

---

## 5.6 GATHERING REQUIREMENTS OF SYSTEM

---

Finalizing the requirements of the system to 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 the 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.

### 5.6.1 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 organising 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 the session. IT staff listen and take notes regarding issues and requirements mentioned by the clients and analysts. They can contribute their ideas related to technical limitations of the 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 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 the 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.

## **5.6.2 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 the 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 the 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 in cases where data has to be shared.
3. Important issues, such as security and validation, are not given importance

### **Check Your Progress 3**

1. .... is nothing but a model of the software to be built.
2. In most of the projects, missing the delivery schedules will lead to .....
3. .... is a process in which group meetings are held to analyse the problem and define the requirements of the desired system.

---

## **5.7 SUMMARY**

---

The process of systems planning is a critical activity in the life of a project. Here, we have focused on determination of requirements, gathering of information about the existing system. There are many techniques for requirements determination which include interviews, questionnaires, group discussions, site visits, and presentations. One or more of the above techniques are used to gather adequate information about the current system. Each technique has its own advantages and disadvantages. In personal interview, the systems analyst gathers information through face to face interaction. It is very common and simple method of fact finding. In a group discussion, a group of individuals is called from different work groups. In this method, problems of all the sections are discussed and a suitable and acceptable solution is arrived at. In the process of site visits, the systems analyst watches the activities and learns about the system. Questionnaires are special type of documents which allow the system analyst to collect information from the respondent.

In this unit, the process of study of feasibility of developing the system is examined. In feasibility study, it is stated whether the project assessment can be accepted for development or is to be rejected for its infeasibility. The key activity in the project planning is the assessment of different feasibility issues associated with the project. It includes economic, technical, operational and legal issues. The economic feasibility judges the cost effectiveness of the project. There are two types of costs involved in a project:

- System Development Costs.
- Operation and Maintenance Costs.

The benefit consists of saving the tangible costs by using the system and the intangible costs by improving the quality of service. In operational feasibility, systems analyst assesses the degree to which the proposed system solves business problem or takes advantage of business opportunity. The legal issues to be considered are copyright law, antitrust legislation, foreign trade legislation, etc.

There are several modern information gathering techniques used by the systems analyst. Some of them are: Joint Application Development (JAD) and Prototyping. JAD is a structured process in which users, managers, and analysts work together through a series of meetings to specify system requirements.

---

## **5.8 SOLUTIONS/ANSWERS**

---

### **Check Your Progress 1**

1. Structured interviews, Unstructured interviews.
2. Questionnaires.
3. Development.

### **Check Your Progress 2**

1. Feasibility Study.
2. Operational Feasibility.
3. Costs and benefits.

### **Check Your Progress 3**

1. Prototype.
2. Penalties.
3. Joint Application Development.

---

## **5.9 FURTHER READINGS**

---

- Jeffrey A.Hoffer, Joey F.George and Joseph S.Valacich;  
*Modern Systems Analysis and Design*; Pearson Education; Third Edition; 2002.
- Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *Systems Analysis and Design Methods*; Tata McGraw Hill; Fifth Edition;2001.
- Elias M. Awad; *Systems Analysis and Design*; Galgotia Publications;  
Second Edition; 1994.
- Perry Edwards; *Systems Analysis and Design*;McGraw Hill Publication;1993.

### **Reference Websites**

<http://www.rspa.com>

<http://www.cbpa.csusb.edu/flin/info609/sysplan>

---

## UNIT 6    MODULAR AND STRUCTURED DESIGN

---

Structure	Page Nos.
6.0 Introduction	17
6.1 Objectives	17
6.2 Design Principles	18
6.2.1 Top Down Design	
6.2.2 Bottom Up Design	
6.3 Structure Charts	20
6.4 Modularity	23
6.4.1 Goals of Design	
6.4.2 Coupling	
6.4.3 Cohesion	
6.5 Summary	29
6.6 Solutions/Answers	29
6.7 Further Readings	30

---

### 6.0 INTRODUCTION

---

In this unit, you will learn the process of designing system's internals. We will begin at an abstract level, taking system's processes, in the form of data flow diagrams, and convert them to structure charts. Structure charts represent design graphically. You will learn the process of drawing structure charts and how to derive them from dataflow diagrams. The structure charts you create will form the basis for the structure of the system you design and build. Decisions made at this point will influence the overall design and implementation of the information system (IS). So, you need to be aware of what constitutes a good design. You will learn about some guidelines that will help you achieve it. The primary goal behind good design is to make your system easy to read and easy to maintain. The primary way to do this is to divide the problem solutions into smaller and smaller pieces or modules. The smaller the pieces or modules the easier it is to program, to read and to revise due to changing users' requirements. Modularisation, therefore promotes ease of coding and maintenance. On the other hand, modularisation is not simply reduction of size but it is also a reflection of function that is what particular piece of a system i.e. a module supposed to do. One guideline for good design is to maximize cohesion, the extent to which a part of the system is designed to perform one and only one function or task. Modules that perform single task are easier to write and maintain than those performing different tasks. As modularisation also involves how different parts of the system work in conjunction with each other, another guideline for good design is to minimize coupling, the extent to which different parts of the system are dependent on each other.

---

### 6.1 OBJECTIVES

---

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

- know the meaning of design;
- learn the process of top down design and bottom up design;
- learn the process of drawing a structure chart;
- learn the goals of design;
- differentiate between five types of coupling and apply them in programs; and
- differentiate between seven types of cohesion and apply them in programs.

Design bridges the gap between specifications and coding. The design of the system is correct if the system is built according to the design that satisfies the requirements of that system.



Some of the properties of design are:

**Verifiability:** It is concerned with how easily the correctness of design can be argued.

**Traceability:** It helps in design verification. It requires that all design must be traceable to the requirements.

**Completeness:** The software must be complete in all respects.

**Consistency:** It requires that there are no inherent inconsistencies within the system.

**Efficiency:** It is concerned with proper usage of resources by the system.

**Simplicity:** It is related to simple design so that user can easily understand and use it. Simple designs are easy to maintain in the long run or through the lifecycle of a software system.

---

## 6.2 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.

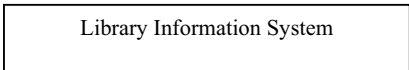
### 6.2.1 Top Down Design

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:

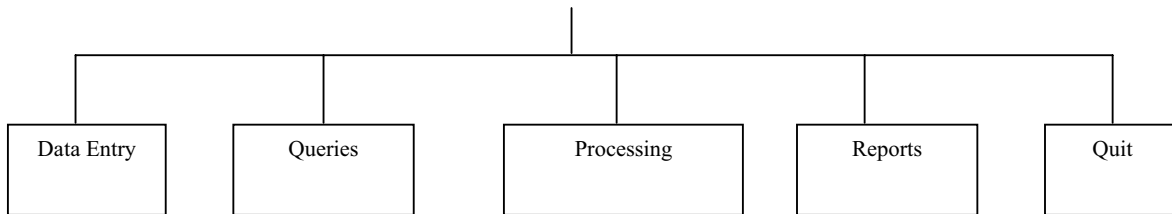


Library Information System

Figure 6.1: The top (root) of software system

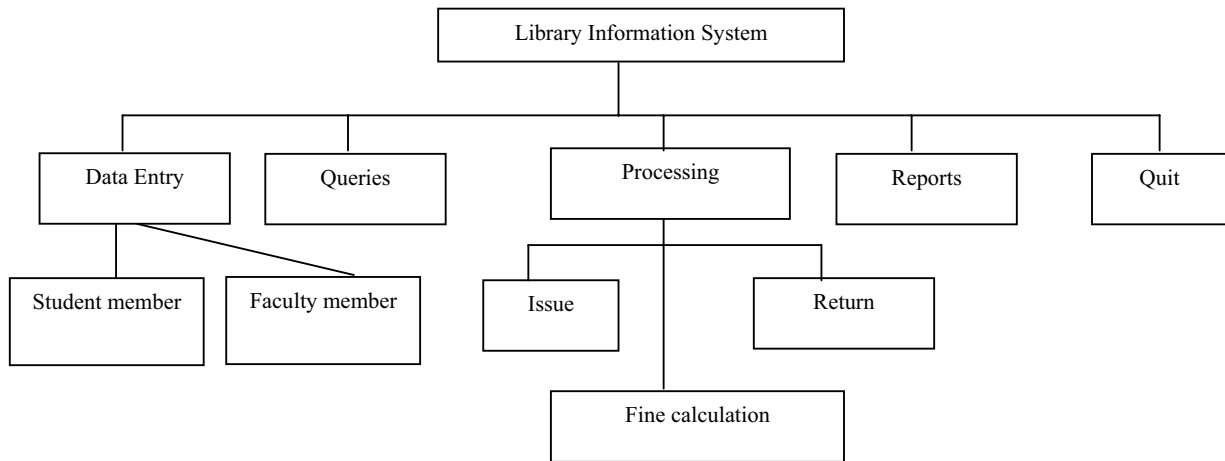


Library Information System



**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 the scratch. We can always start from the main menu and proceed down the hierarchy designing data entry modules, queries modules, etc.

### 6.2.2 Bottom Up Design

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 the 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.

### Check Your Progress 1

1. .... 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.
2. .... 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.

3. .... starts from the bottom and move upwards towards the top of the software.

## 6.3 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. It's subordinate modules are *Get X* and *Make Y*. The subordinate modules of *Get X* are *Get W* and *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 realised 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.

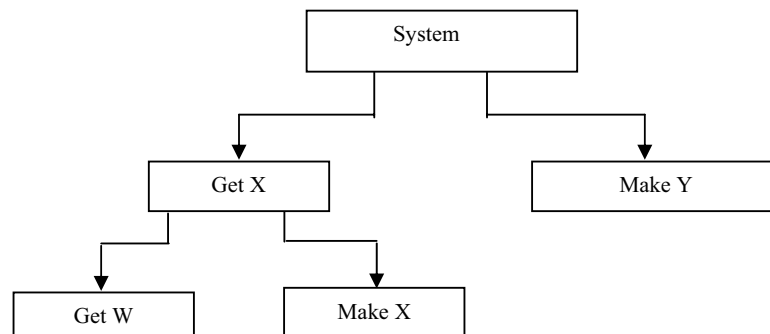


Figure 6.4: Hierarchy of a Structure Chart

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 the 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 within the module. Figure 6.7 shows the symbol for module.

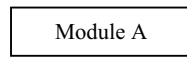


Figure 6.7: Symbol for Module

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

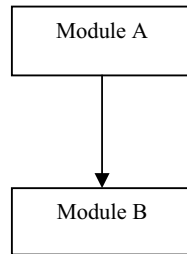


Figure 6.8: Symbols for superordinate module A and subordinate module B

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

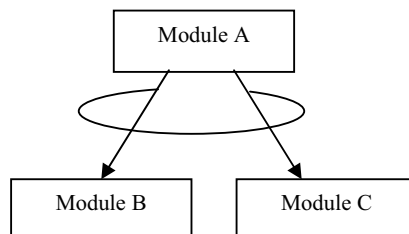


Figure 6.9: Repeated calls of Modules B and C by Module A

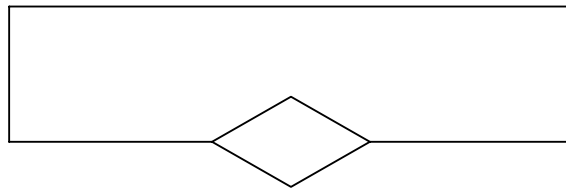


Figure 6.10: Subordinate modules are called on condition

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 indicate 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.

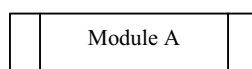


Figure 6.11: Predefined module A

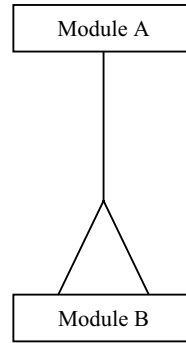


Figure 6.12: Module B is embedded in Module A

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 the 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 the result R to system. Finally, the system module calls Put Result R module and passes the result to it.

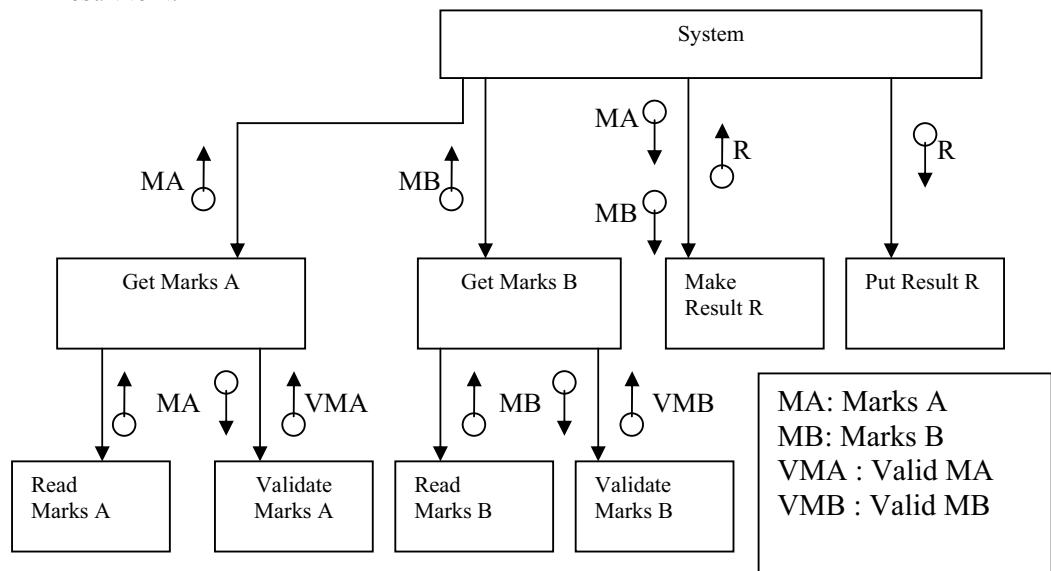


Figure 6.13: Reading a Structure chart

### Check Your Progress 2

1. A ..... depicts the modular organization of an information system.
2. If a menu based system is concerned, the main menu may be considered as the ..... and the menu options in it may be considered as subordinate modules.
3. Modules at ..... don't call any other modules.

## 6.4 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.

### 6.4.1 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 or 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 is 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 the lines of code, easier to read.
- A module should not perform more than one function. There should be no line 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’s function should be describable easily in a few words. This is called *cohesion*.
- Modules at the lower level of the design are called by more than one superordinate module. It means that multiple superordinate modules use most of the modules at the lower level.

### 6.4.2 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-1 is largely dependent on another module-2, then, any error in module-2 will affect the functionality of module-1. 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 changes 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.

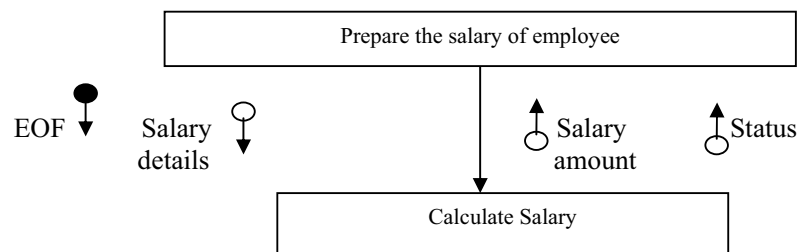


Figure 6.14: Example of Data Coupling

**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 the 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.

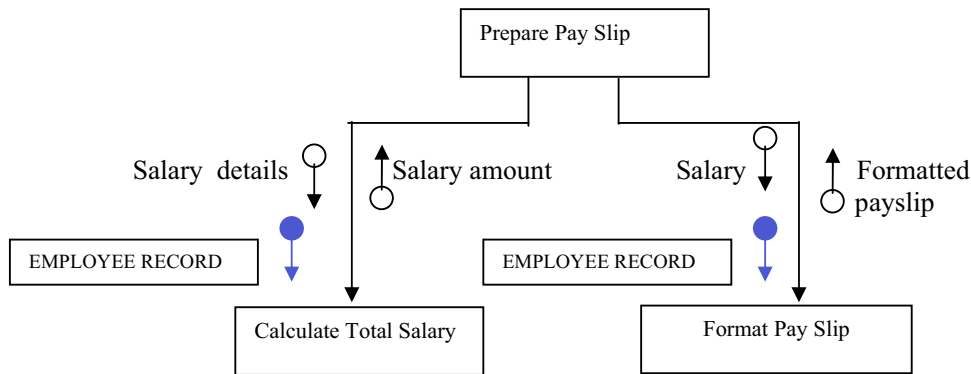


Figure 6.15: Example of Stamp Coupling

**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.

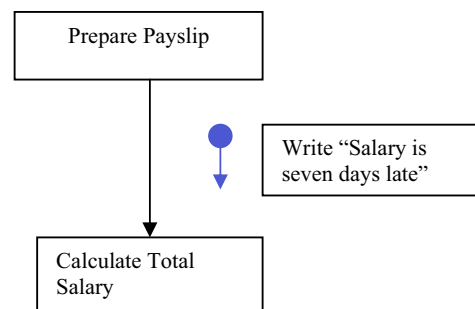


Figure 6.16: Example of Control Coupling

**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 the High level languages. A variable can be declared at appropriate level so that it is 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.

### 6.4.3 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 the 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 inventory,  
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.

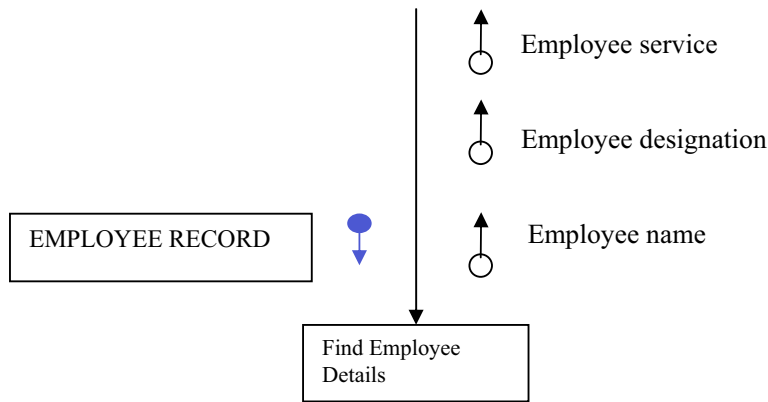


Figure 6.17: An example of a Communicational Cohesion module

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 and 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 modules instead of one communicational cohesive module in Figure 6.17.

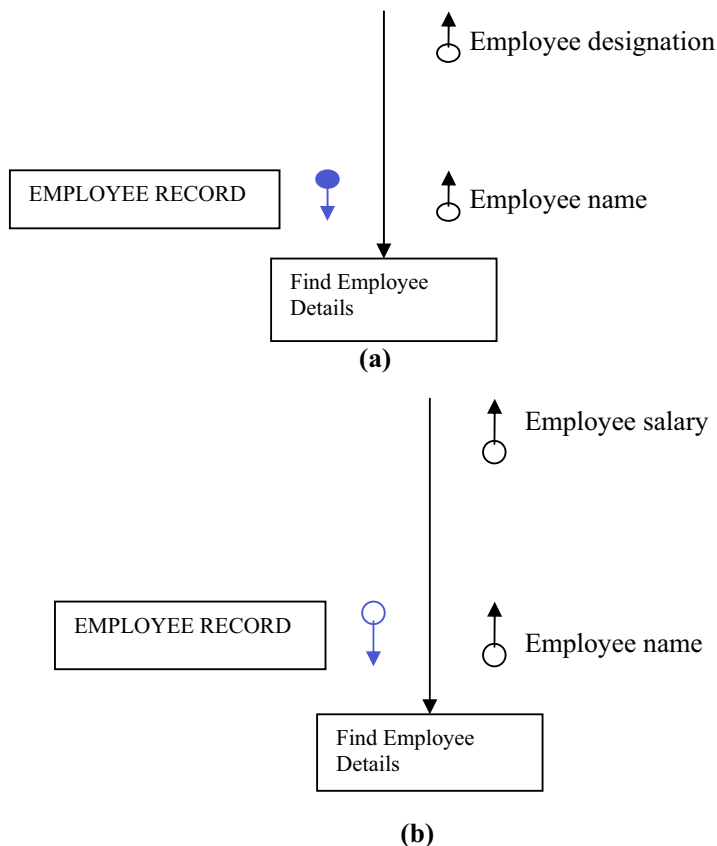


Figure 6.18 (a) & (b): An example of two functionally cohesive modules which resulted due to the split of one communicational cohesive module

**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 the 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, the 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.

**Check Your Progress 3**

1. A module should not be assigned the duty of performing more than ..... function.
2. The coupling between the modules should be .....
3. .... is the best and .... is the worse type of cohesions.

---

**6.5 SUMMARY**

---

This unit focussed on the requirements of a good design. We have studied various principles of good design. The two design techniques, namely, Top Down Design and Bottom Up Design have been explained. The process of depicting the modular organization of a system has been explained using Structure charts. Different symbols used in Structure charts have been discussed. An example structure chart for reading the marks in various courses and computing the result has been demonstrated. The issue of the degree of communication that should be present between various modules has been discussed through Coupling and Cohesion. There are 5 types of coupling namely Data coupling, Stamp coupling, Control coupling, Common coupling and Content coupling. There are a total of 7 types of cohesion namely, Functional cohesion, Sequential cohesion, Communicational cohesion, Procedural cohesion, Temporal cohesion, Logical cohesion and Coincidental cohesion.

---

**6.6 SOLUTIONS/ANSWERS**

---

**Check Your Progress 1**

1. Abstraction
2. Top Down Design
3. Bottom Up Design

**Check Your Progress 2**

1. Structure Charts
2. Coordinating module , Subordinate modules
3. Lower level

**Check Your Progress 3**

1. One
2. Minimum
3. Functional cohesion, Coincidental cohesion

---

**6.7 FURTHER READINGS**

---

Joey George, J. Hoffer and Joseph Valacich; *Modern Systems Analysis and Design*; Pearson Education; Third Edition; 2001.

Alan Dennis, Barbara Haley Wixom; *Systems Analysis and Design*; John Wiley & Sons; 2002.

**Reference Websites**

<http://www.rspa.com>

---

## UNIT 7    SYSTEM DESIGN AND MODELING

---

Structure	Page Nos.
7.0 Introduction	31
7.1 Objectives	31
7.2 Logical and Physical Design	31
7.3 Process Modeling	36
7.3.1 Data Flow Diagrams	
7.4 Data Modeling	38
7.4.1 E-R Diagrams	
7.5 Process Specification Tools	39
7.5.1 Decision Tables	
7.5.2 Decision Trees	
7.5.3 Structured English Notation	
7.6 Data Dictionary	43
7.7 Summary	44
7.8 Solutions/Answers	44
7.9 Further Readings	47

---

### 7.0 INTRODUCTION

---

System Design is the specification or construction of a technical, computer based solution for the business requirements identified in system analysis phase. During design, system analysts convert the description of the recommended alternative solution into logical and then physical system specifications. S/he must design all aspects of the system from input and output screens to reports, databases, and computer processes. Databases are designed with the help of data modeling tools (for example, E-R diagrams) and computer processes are designed with the help of Structured English notation, Decision Trees and Decision Tables. Logical design is not tied to any specific hardware and software platform. The idea is to make sure that the system functions as intended. Logical design concentrates on the business aspects of the system. In physical design, logical design is converted to physical or technical specifications. During physical design, the analyst's team takes decisions regarding the programming language, database management system, hardware platform, operating system and networking environment to be used. At this stage, any new hardware or software can also be purchased. The final output of design phase is the system specifications in a form ready to be turned over to programmers and other system builders for construction (coding).

---

### 7.1 OBJECTIVES

---

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

- design major components of the system;
- identify tools for every component;
- to learn the process of using tools; and
- integrate component designs into a whole system specification.

---

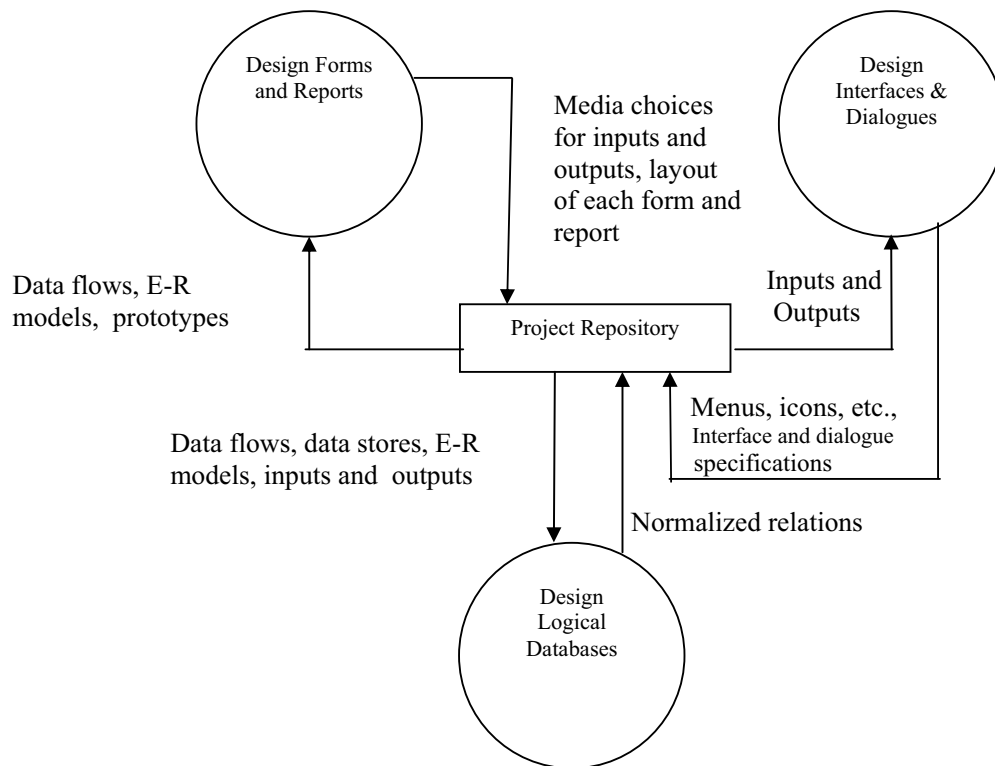
### 7.2 LOGICAL AND PHYSICAL DESIGN

---

**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.

### **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 GUI (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.

### 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 reports—Present information with little or no filtering or restrictions;  
Summary reports—Categorize information for managers who do not want to go through details; and

Exception reports—Filter data before it is presented to the manager as information.

- 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).

### **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.

### 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 entity. 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.

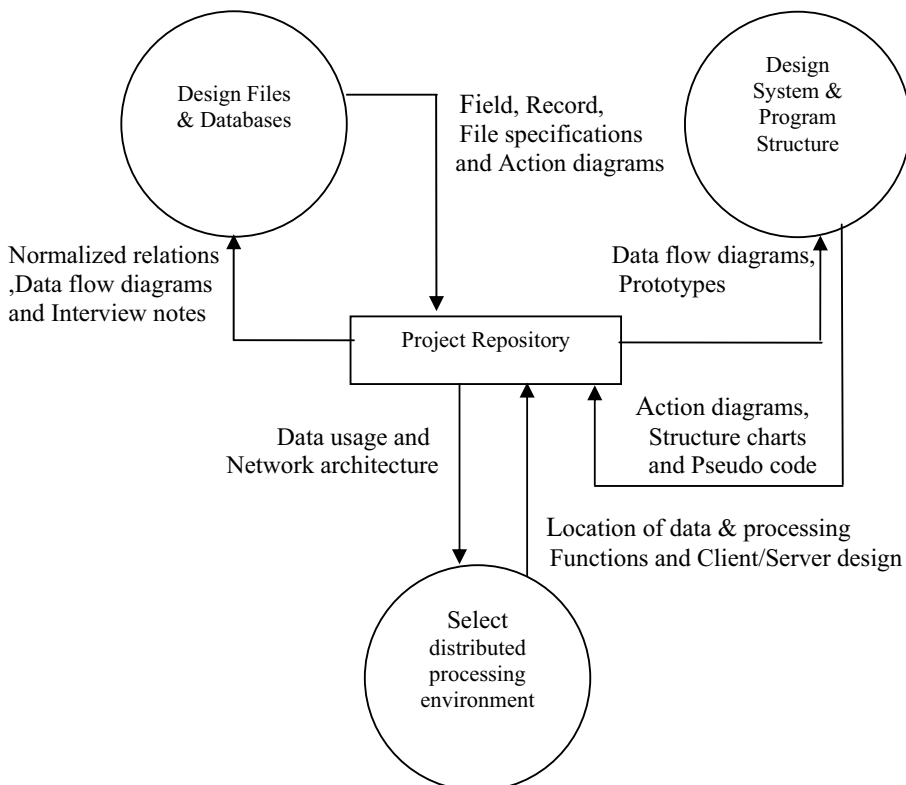


Figure 7.2: Steps in Physical Design

---

## 7.3 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.

### 7.3.1 Data Flow Diagrams

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 create 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.

The following are various components of a Data Flow Diagram:

#### 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



#### 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. 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



#### 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



#### 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 a 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 the 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.

#### Check Your Progress 1

1. Develop a context level DFD and First level DFD for the hospital pharmacy system describe in the following case study: “The pharmacy at Sanjeevni Hospital fills medical prescriptions for all patients and distributes these medications to the nurse stations responsible for the patient’s care. Medical prescriptions are written by doctors and sent to the pharmacy. A pharmacy technician reviews the prescriptions and sends them to the appropriate pharmacy

station. At each station, a pharmacist reviews the order, checks the patient file to determine the appropriateness of the prescriptions and fills the order. If the pharmacist does not fill the order, the prescribing doctor is contacted to discuss the situation. In this case the order may ultimately be filled or the doctor may write another prescription, depending on the outcome of the discussion. Once filled, a prescription level is generated listing the patient's name, the drug type and dosage, an expiration date and any special instruction. The level is placed on the drug container and the order is sent to the appropriate nurse station. The patient's admission number, the drug type, and the cost of the prescription are then sending to the billing department".

---

## 7.4 DATA MODELING

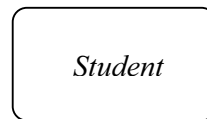
---

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.

### 7.4.1 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.

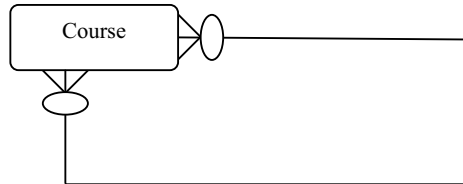


Figure 7.3: Example of Recursive relationship

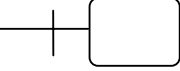
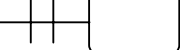
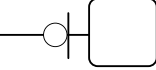
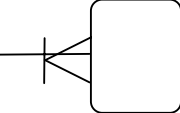
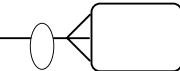
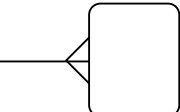
Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one (One and only one)	1	1	 or 
Zero or one	0	1	
One or more	1	Many (>1)	
Zero, one or more	0	Many (>1)	
More than one	>1	>1	

Figure 7.4: Different types of cardinality

## Check Your Progress 2

1. Draw an E-R diagram for the following case study:  
 “In a purchasing department at one company, each purchase request is assigned to a “case worker” within the purchasing department. This caseworker follows the purchase request through the entire purchasing process and acts as the sole contact person with the person or unit buying the goods or services. The purchasing department refers to its fellow employees buying goods and services as “customers”. The purchasing process is such that purchase request must go to vendors. The product or service can simply be bought from any approved vendor, but the purchase request must still be approved by the purchasing department.”

## 7.5 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.

### 7.5.1 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.

Rules									
Process Name		1	2	3	4	5	6	7	8
Conditions	_____	X	—	.	.	.	.	.	.
	_____								
	_____								
	_____								
Actions	_____	—	X	?	.	.	.	.	.
	_____								
	_____								
	_____								

X— Action (condition is true)

— Condition is irrelevant for this rule

? — Unknown rule

Figure 7.5: An example Decision Table

### 7.5.2 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.

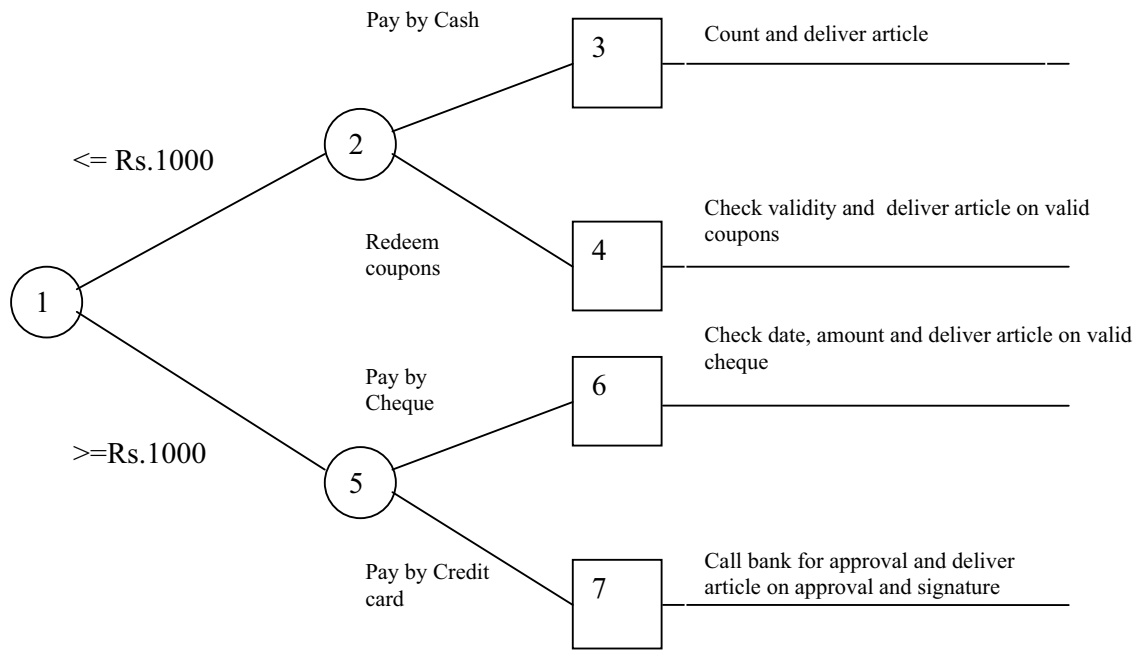


Figure 7.6: An Example Decision tree for a Customer Bill Payment System

### 7.5.3 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 of 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 operates.  
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
Subtract (-)	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 1 sentences are executed. If it is false, then group 2 sentences are executed.
- b) A structure which allows a choice between any number 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 group 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 conditions & 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

### Check Your Progress 3

1. Draw a decision table for following policy statement:

“A bank offers two types of savings accounts, regular rate and split rate. The regular rate account pays dividends on the account balance at the end of each quarter. Funds withdrawn during the quarter earn no dividends. There is no minimum balance on the regular account. Regular rate account may be insured. Insured account gets 5.75 percent annual interest. Uninsured regular rate accounts get 6.00 percent annual interest.

For split rate accounts, dividends are paid monthly on the average daily balance for that month. Daily balances go up and down in accordance with deposits and withdrawals. The average daily balance is determined by adding each days closing balance and dividing this sum by the number of days in the month.

If the balance dropped below Rs.2000/- during month then no dividend is paid. So, if the average daily balance is less than Rs.2000/-, then no dividend is paid. Otherwise, if the average daily balance is Rs.2000/- or more, then an interest of 6% per annum is paid on the first Rs.5000/-, 6.5% on the next Rs.20000/- and 7% on funds over Rs.25000/-. There is no insurance on split rate.”

2. Draw a Decision Tree for the policy statement stated above in question no.1.

---

## 7.6 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 without accessing data dictionary. If analysts want to know the 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 is not tampered.

Analysts use data dictionaries for the 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 omissions 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.

---

## 7.7 SUMMARY

---

Design is the phase that precedes coding. All the components of system are designed logically. Then physical aspects of the system are designed. Form design, report design, database design and program design etc. are designed with the help of GUI controls, process modeling tools, data modeling tools and process specification tools. These tools reduce the complexity of the design process. A Data Dictionary is data about data. These elements centre around data and the way they are structured to meet user requirements and organization's needs.

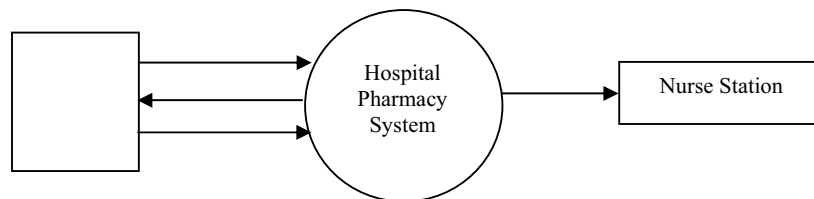
---

## 7.8 SOLUTIONS/ANSWERS

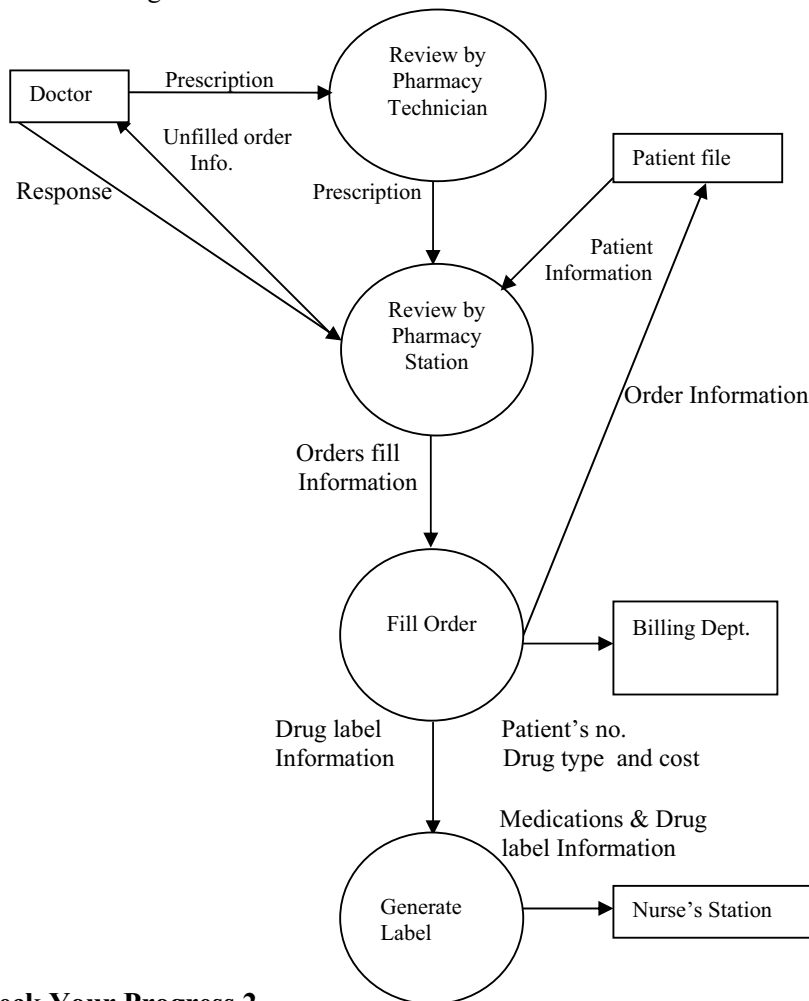
---

### Check Your Progress 1

1. The following is the context level DFD:

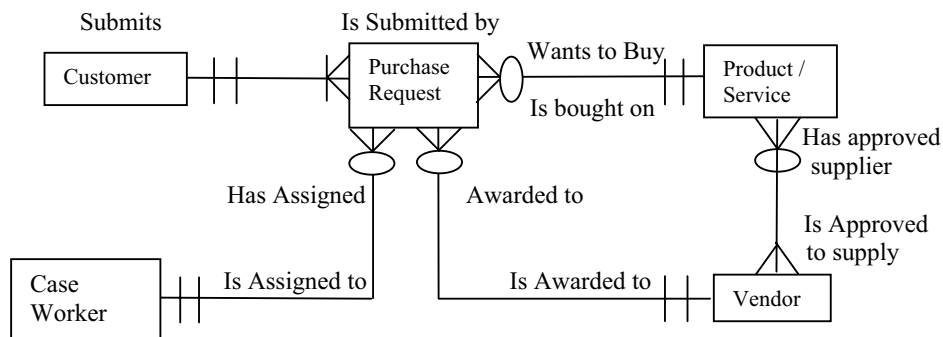


The following is the first level DFD:



## Check Your Progress 2

- The following is the ER diagram:



## Check Your Progress 3

- We have to identify conditions and values.

### Data Elements Condition

- Account type
- Insurance

### Values

R = Regular  
S = split  
Y = Yes  
N = No

3. Balance Dropped below Rs.2000/- during month?

Y = Yes

### Steps for Decision Table

To identify conditions (Data Elements) and their values.

1. To determine the maximum number of rules. The maximum number of rules in a decision table is calculated by multiplying the number of values for each condition data element.

Example: Condition 1 offers two values

Condition 1 offers two values

Condition 1 offers two values

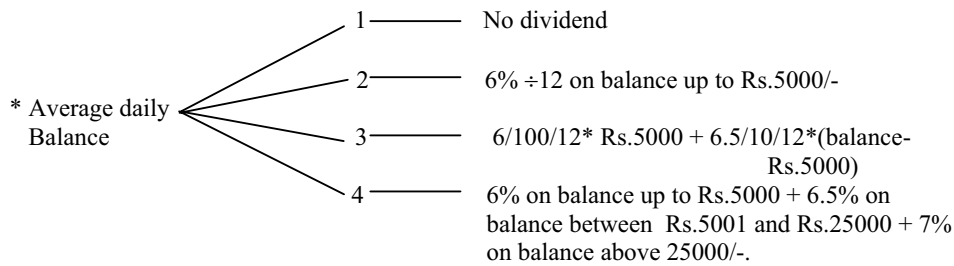
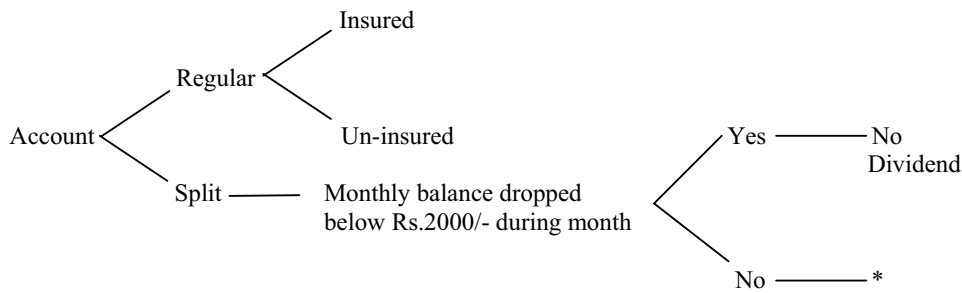
then, total number of rules =  $2 \times 2 \times 2 = 8$

2. To identify the possible actions. It means to identify each independent action to be taken for the decision or policy.
3. To enter all possible rules and record all conditions and actions in their respective places in the decision table.
4. The way of defining rules:
  - a. For the first condition, to alternate its possible values Example: If Y and N are two possible values and there are 8 rules in the table then we will define these rules as:  
Y N Y N Y N Y N
  - b. For condition two,  
size of pattern that repeats in step (a) i.e. 2  
 $\Rightarrow$  Y Y N N Y Y N N
  - c. For condition three,  
size of pattern that repeats in step (b) i.e., 4 so the possible combination is:  
Y Y Y Y N N N N
5. After arranging all conditions, actions and rules  
Now action will be taken according to rule i.e.
  - X : Action (correct rule)
  - : Irrelevant or indifference symbol
  - ? : Unknown rule
6. To verify the policy. Analyst can resolve any rules for which the actions are not specified. Analyst can also resolve apparent contradictions such as one rule with two possible actions.
8. Finally, Analysts must simplify the decision table:
  - \* To eliminate impossible rules.
  - \* The rules can be consolidated into a single rule for indifferent conditions where indifferent condition is a condition whose values do not affect the decision and always result in the same action.

The resultant Decision Table is given below:

Process Name		Dividend rate Rules						
		1	2	3	4	5	6	7
Conditions	Account type	R	R	S	S	S	S	S
	Insurance	Y	N	--	--	N	N	N
	Balance dropped below Rs.2000/- during month	--	--	Y	N	N	N	N
	Average daily balance	--	--	--	1	2	3	4
Actions	Pay no dividend			X	X			
	5.75% ÷ 4 quarterly dividend on entire balance.	X						
	6.000% ÷ 4 quarters		X					
	6.000% ÷ 12 monthly dividend on balance up to Rs.5000/-					X	X	X
	6.500% ÷ 12 monthly dividend on balance between Rs.5001/- to Rs.20000/-						X	X
	7.000% ÷ 12 monthly dividend on a balance of above Rs.25000/-							X

2.



## 7.9 FURTHER READINGS

Jeffrey L. Whitten, Lonnie D. Bentley and Kevin C. Dittman; *Systems Analysis and Design Methods*; Tata McGraw Hill Publishing Company Limited; Fifth Edition; 2000.

Jeffrey A. Hoffer, Joey F. George and Joseph S. Valacich; *Modern Systems Analysis and Design*; Pearson Education Publishing Company Limited; Third Edition; 2001.

V. Rajaraman; *Analysis and Design of Information Systems*; Prentice-Hall of India Private Limited; Second Edition.

### Reference Websites

<http://www.rspa.com>

<http://www.ieee.org>

---

## UNIT 8 FORMS AND REPORTS DESIGN

---

Structure	Page Nos.
8.0 Introduction	5
8.1 Objectives	6
8.2 Forms	7
8.2.1 Importance of Forms	
8.3 Reports	8
8.3.1 Importance of Reports	
8.4 Differences between Forms and Reports	9
8.5 Process of Designing Forms and Reports	10
8.6 Deliverables and Outcomes	10
8.7 Design Specifications	
8.7.1 Narrative Overviews	
8.7.2 Sample Design	
8.7.3 Testing and Usability Assessment	
8.8 Types of Information	11
8.8.1 Internal Information	
8.8.2 External Information	
8.8.3 Turnaround Documents	
8.9 General Formatting Guidelines	15
8.9.1 Meaningful Titles	
8.9.2 Meaningful Information	
8.9.3 Balanced Layout	
8.9.4 Easy Navigation	
8.10 Guidelines for Displaying Contents	16
8.10.1 Highlighting Information	
8.10.2 Using Colour	
8.10.3 Displaying Text	
8.10.4 Designing Tables and Lists	
8.11 Criteria for Form Design	17
8.11.1 Organization	
8.11.2 Consistency	
8.11.3 Completeness	
8.11.4 Flexible Entry	
8.11.5 Economy	
8.12 Criteria for Report Design	18
8.12.1 Relevance	
8.12.2 Accuracy	
8.12.3 Clarity	
8.12.4 Timeliness	
8.12.5 Cost	
8.13 Summary	20
8.14 Solutions/ Answers	20
8.15 Further Readings	22

---

### 8.0 INTRODUCTION

---

This unit deals with the interface of software with users. Usually, the interface is through forms and reports that are associated with the system. In this unit, we will study different aspects of designing Forms and Reports, as these are the key ingredients for successful systems. As the quality of a system greatly depends upon the quality of inputs and outputs, the process of designing forms and reports is very important.

The logical phase within the system development life cycle (SDLC) deals with the issues related to the design of system inputs and outputs (forms and reports) as shown in figure 8.1. Forms are used to collect data for the system and reports to deliver information to users. With forms, data can be entered into the database. With data entered in the database, it is possible to use a query language so as to generate reports

about the data. In this unit, we shall also look into the deliverables produced during the process of designing forms and reports. Formatting of forms and reports is also discussed as this serves as the building block for designing.

Forms and reports should be well conceived and attractive in design. In order to achieve this goal, we shall look into different criteria that are to be followed while designing forms and reports.

---

## 8.1 OBJECTIVES

---

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

- define Forms & Reports and their importance in real life;
- list the process of designing Forms & Reports ;
- know about Internal information, External information, Turnaround documents and differentiate between them;
- apply the general guidelines for formatting Forms and Reports; and
- specify different criteria for designing Forms and Reports.

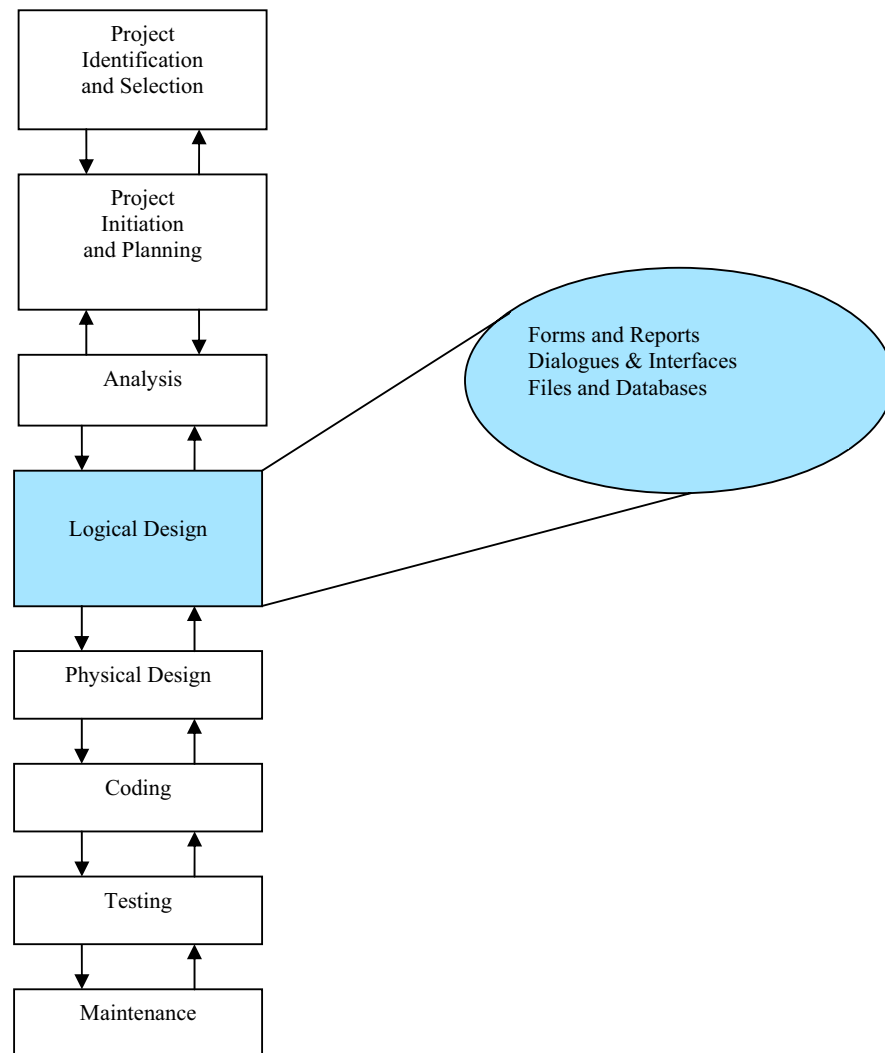


Figure 8.1: Systems Development Life Cycle with Logical Design Phase Highlighted

---

## 8.2 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 than paper forms.

Examples of forms are Business forms, Electronic spread sheet, ATM transaction layout, etc.

Figure 8.2 shows a simple form that is used to collect employee details.

**EMPLOYEE DETAILS**

**Name**

Choose: **Title** ▾

First name:

Last name:

**Contact details**

Email address:

**Address**

Street 1:

Street 2:

Town:

Figure 8.2: A Simple Form

### 8.2.1 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.

---

## 8.3 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 the employees in the organization.

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

### 8.3.1 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.

---

## 8.4 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.

---

## 8.5 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 task will the users be 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 tool or the standard development tools that are used within the organization be used. Usually, initial prototype 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 now a days Online graphical tools for designing forms and reports are very much in use in most professional development organizations.

## Check Your Progress 1

1. Define Form and Report. Also, differentiate between them.  
.....  
.....  
.....  
.....
2. List the advantages of using Forms and Reports.  
.....  
.....  
.....  
.....
3. Describe the process of designing Forms and Reports.  
.....  
.....  
.....  
.....
4. State True or False for the following:
  - a) A form is a type of layout that typically lists and summarizes data from several unrelated data base records.
  - b) A report is a type of output that typically lists data associated with one data base record.
  - c) Within reports, we can include sub-forms and sub-reports.

---

## 8.6 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.

---

## 8.7 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.

### 8.7.1 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.

### 8.7.2 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.

### 8.7.3 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.

---

## 8.8 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.

### 8.8.1 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 can 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
<div>Return To Summary</div> <div>Close</div>					

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
<div>View additional Reports</div> <div>Close</div>				

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

### 8.8.2 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.

### 8.8.3 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

## 8.9 GENERAL FORMATTING GUIDELINES

Proper formatting of forms and reports is very much essential. But, unfortunately, a definitive set of rules for delivering every type of information to users is yet to be defined and these rules are continuously evolving along with the rapid changes in

technology. However, certain guidelines are available which are to be considered while formatting information. One of the most important thing to keep in mind while designing usable forms and reports is that there should be active interaction with the users. If the appearance of forms or reports is awkward to use or confusing, the users will be dissatisfied even if the rest of the system performs well. Formatting of forms and reports influences individual task performance and perceptions of usability.

The following are general guidelines for the design of Forms and Reports, which make the Form, or Report more acceptable:

- Meaningful titles,
- Meaningful information,
- Balanced layout, and
- Easy navigation.

### 8.9.1 Meaningful Titles

The form or report should contain title that is clear and specific. It should clearly describe the content and use of form or report. It should also include the date on which the form or report was generated. Page heading formats should be consistent throughout the system. For example, the date should always appear in the same place. Column headings should clearly indicate the contents of the columns and should be separated from the body using extra blank lines, horizontal rule etc.

### 8.9.2 Meaningful Information

Only the information that is relevant and needed by the user should be displayed on the form or report. Information should be provided in such a manner that the user could use it without any modification. All the information irrelevant to the intent of the form or report should not be displayed.

### 8.9.3 Balanced Layout

The information should be balanced on the screen or page, i.e., the display should not be too crowded and not too spread out. When deciding where to put individual fields on the form or report, we should see that the form or report is easy to understand and to use. The most important information should be placed where they are easiest to find (generally, at the beginning). The different fields should be separated by means of extra spaces whenever possible so that a subsequent field expansion will not necessarily force to redesign the entire layout. All related information should be grouped together wherever possible. For example, name, street address and city/state/pin code can be grouped together. Appropriate line spacing greatly enhances the readability of a form or report. We should insert extra blank lines to indicate where headers end and the body of information begins, where one multi-line detail ends and the next begins, where one group of items end and another begins, etc.

### 8.9.4 Easy Navigation

It should be possible for the user to easily move forward and backward through the contents of form or report. At any instance, it should be possible for the user to know where exactly s/he is (e.g., Page 2 of 3). The user should be notified when s/he is on the last page of a multi-page sequence. The user must be able to exit or quit the report or form easily.

### Check Your Progress 2

1. What do you mean by internal information, external information and turnaround document?

.....  
 .....



- .....
- .....
2. List the general formatting guidelines.

- .....
- .....
- .....
- .....
3. What are the different sections of design specifications?

- .....
- .....
- .....
- .....
4. State True or False for the following:

- a) Balance is a way to draw attention to or away from data in output.
- b) Turnaround documents are produced for people outside the organization, but ultimately return as input to the internal system.
- c) Narrative overview contains the general guidelines for formatting forms and reports.

---

## 8.10 GUIDELINES FOR DISPLAYING CONTENTS

---

The way the form or a report appears to the human eye has a lot of impact on the user and by following specific guidelines for highlighting information such as using colour to display text, and presenting numeric tables and lists, we can make the form or report more presentable.

### 8.10.1 Highlighting Information

Highlighting the information will enhance the appearance of the output. However, highlighting should be used sparingly to draw the user to or away from certain information and to group together related information. Highlighting of information can be carried out using different methods such as using blinking and audible tones, colour difference, intensity difference, font and size differences, underlining, etc. Highlighting methods should be selected and consistently used based upon the level of importance of the emphasized information.

Highlighting will be very useful in situations such as the following:

- Notifying users of errors in data entry or processing;
- Drawing attention to high priority messages; and
- Providing warnings to users in situations like unusual data values or an unavailable device.

### 8.10.2 Using Colour

Colour influences the usability of the system. Use of appropriate colours while designing has several advantages which are given below:

- Soothes or strikes the eye;
- Emphasizes the logical organization of information;
- Draws attention to warnings;
- Evokes more emotional reactions; and
- Use of colours in graphs and charts helps in better understanding.

The following are some disadvantages of using colours:

- Resolution may degrade with different displays;
- Colour fidelity may degrade on different displays; and
- Printing or conversion to other media may not be possible easily.

### 8.10.3 Displaying Text

Now a days, as the text based applications such as e-mail, bulletin boards , chatting, etc., are being widely used, textual output is becoming increasingly important. Some of the guidelines to be followed in order to display text are given below:

We should use appropriate punctuation wherever required. The text should be properly spaced and there should be blank line between paragraphs. We should not hyphenate words between lines or use obscure abbreviations and acronyms while displaying textual information.

### 8.10.4 Designing Tables and Lists

The content and meaning of tables and lists are significantly derived from the format of the information. There are a few simple guidelines that are to be followed while designing tables and lists. Use meaningful labels to all columns and rows and separate labels from other information by using highlighting. The data displayed should be sorted in a meaningful order (like ascending or descending or alphabetic). Columns should be separated by at least two spaces between them. We should make use of single typeface except for emphasis.

Numeric, textual and alphanumeric data should be properly formatted. For example, numeric data should be right justified and columns should be aligned using decimal points or other delimiter.

Textual data should be left justified and line length should be somewhere between 30-40 characters per line. If there are long sequences of alphanumeric data, then they should be broken into small groups of three to four characters each.

---

## 8.11 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.

### 8.11.1 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.

### 8.11.2 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.

### **8.11.3 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.

### **8.11.4 Flexible Entry**

It should be possible to enter data by hand or with a typewriter. In most cases, both kinds of entries occur.

### **8.11.5 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.

---

## **8.12 CRITERIA FOR REPORT DESIGN**

---

Reports convey information from one or 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.

### **8.12.1 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.

### **8.12.2 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.

### **8.12.3 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.

#### 8.12.4 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.

#### 8.12.5 Cost

Every report has two costs. First is the cost of preparation, which consists of analysis, design, computation and distribution. Second is the 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.

### Check Your Progress 3

1. When can highlighting be used to convey special information to users?  
.....  
.....  
.....  
.....
2. What are the advantages and disadvantages of using colours when designing system outputs?  
.....  
.....  
.....  
.....
3. Specify the criteria used to identify a good report design.  
.....  
.....  
.....  
.....
4. Specify the criteria used to identify a good form design.  
.....  
.....  
.....  
.....
5. Describe how numeric, textual and alphanumeric data should be formatted in a table or a list.  
.....  
.....

---

## 8.13 SUMMARY

---

As we have seen, the main aim of this unit is to help the reader in selecting appropriate formats for conveying information on Forms and Reports. Here, we have seen that forms and reports are instruments of communication between people and computers.

In this unit, we have seen that Forms are used to collect or present information and Reports to convey information on a collection of items. Also, we have seen that information can take any of the three general forms:

Internal information is the information, which is used only within the organization.

External information is the information, which is used for distribution outside the organization.

Turnaround documents are documents that are produced for outside parties, but ultimately return (in part or in whole) as an input to internal system.

There are many ways in which information can be collected and formatted. Also, specific guidelines should be followed, as these guidelines will help in creating professional usable systems. Here, we have presented a variety of guidelines covering use of titles, layout of fields, highlighting data, use of colours, navigation between pages, formatting text and numeric data.

We have seen that before we start the designing of forms or reports, we should collect answers for questions like who, what, when, where, and how. Answers to these questions will help us to have a clear idea of what actually is required from the form or report to be designed. After getting answers to above questions, the initial prototype can be created.

A well designed form should satisfy the criteria like Organization, Consistency, Completeness, Flexible entry, Economy and a well design report should satisfy the criteria like relevance, accuracy, clarity, timeliness, cost, etc.

---

## 8.14 SOLUTIONS/ ANSWERS

---

### Check Your Progress 1

1. In computer terminology, a Form identifies the data we want to collect. A report is the information that is organized and formatted to fit the required specification. The difference between Form and Report is that a form is used to collect or present information and reports are used to convey information on a collection of items.
2. Some of the advantages of using the forms are given below:
  - Forms provide an easy way to view data.
  - Using forms, data can be entered efficiently.
  - We can present data in attractive format using forms.
  - Forms offer convenient layout for entering, changing and viewing records present in the database.

The advantages of reports are:

- Reports organize and present data in groups.
- With reports, we can calculate running totals, group totals, grand totals etc.

- We can present data in attractive format.
  - Within reports we can include sub-forms, sub-reports and graphs.
3. While designing Forms and Reports, the following steps have to be taken:
- a) The first is to collect all relevant information regarding the form or Report by asking questions like who, what, when, where and how.
  - b) Refine the above information into an initial prototype.
  - c) Review and evaluate the prototype.
  - d) Design, validate and test the outputs using some combination of prototyping and code generation tools.
4. State True or false
- a) False
  - b) False
  - c) True

### Check Your Progress 2

1. Internal information – For use within the organization  
External information – For distribution outside the organization  
Turnaround documents – Produced for outside parties but are again returned back as input to the Internal system.
2. The general formatting guidelines are:
- Meaningful titles,
  - Meaningful information,
  - Balanced layout, and
  - Easy navigation.
3. The different sections of design specification are:
- Narrative overview
  - Sample design
  - Testing and usability assessment
4. State True or False
- a) False
  - b) True
  - c) False

### Check Your Progress 3

1. Special information such as the following should be conveyed using Highlighting to users:
- Notifying the users in case of some errors,
  - Drawing attention to high priority messages, and
  - Provide any warning messages to users.
2. Advantages of using colours:
- Soothes or strikes the eye,
  - Emphasizes the logical organization of information, and
  - Draws attention to warnings.

Disadvantages of using colours:

- Resolution may degrade with different displays
  - Colour fidelity may degrade on different displays
  - Printing or conversion to other media may not easily translate.
3. Criteria used to identify good report design are:
- Relevance,
  - Accuracy,
  - Clarity,
  - Timeliness, and
  - Cost.
4. Criteria used to identify good form design are:
- Organization,
  - Consistency,
  - Completeness,
  - Flexible entry, and
  - Economy.
5. Numeric data should be right justified and columns should be aligned using decimal point or delimiter. Textual data should be left justified and line length should be somewhere between 30 – 40 characters per line.

---

## 8.15 FURTHER READINGS

---

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

Jeffrey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman; *Systems Analysis and Design Methods*; Tata McGraw Hill; Fifth Edition;2001.

Alan Dennis, Barbara Haley Wixom; *Systems Analysis and Design*;John Wiley & Sons;2002.

K.C.Laudon and J.P.Laudon; *Management Information Systems*; Pearson Education; Seventh Edition.

John W. Satzinger, Stephen D. Burd, Robert B. Jackson; *Systems Analysis and Design in a changing world*; Course Technology Inc.; Third Edition;2004.

### Reference Websites

<http://www.formdesk.com>

<http://www.functionx.com>