Analysts

Customers

Analysis

Design

Interface

Data

Detail

Architecture

PLANNING AND DESIGNING SYSTEMS

2

Indira Gandhi
National Open University
School of Computer and
Information Sciences

MCS-014
SYSTEMS ANALYSIS
AND DESIGN

Block

2

PLANNING AND DESIGNING SYSTEMS

Programme / Course Design Committee

Prof. Sanjeev K. Aggarwal, IIT, Kanpur
Prof. M. Balakrishnan, IIT , Delhi
Prof. Harish Karnick, IIT, Kanpur
Prof. C. Pandurangan, IIT, Madras
Dr. Om Vikas, Sr. Director, MIT
Prof P. S. Grover, Sr. Consultant
SOCIS, IGNOU

Faculty of School of Computer and Information Sciences
Shri Shashi Bhushan
Shri Akshay Kumar
Prof Manohar Lal
Shri V.V. Subrahmanyam
Shri P. Venkata Suresh

Block Preparation Team

Shri M.P.Goel (Content Editor)
Principal Systems Analyst (Retd)
National Informatics Centre
New Delhi

Ms.Huma Anwar
Dept. of Computer Science
Institute of Management & Research
Ghaziabad

Shri K.L.Sarkar
Manager
NSIC
New Delhi

Ms.Tamanna Siddiqui
Dept. of Computer Science
Jamia Hamdard
New Delhi

Shri P. Venkata Suresh
SOCIS, IGNOU

Prof. M.R.Dua (Language Editor)
New Delhi

Course Coordinator : P. Venkata Suresh

Block Production Team

Shri H.K Som, SOCIS

May, 2004

©*Indira Gandhi National Open University, 2004*

ISBN-81-266-1230-4

BLOCK INTRODUCTION

This block is on the planning and designing information systems.

Every organization processes information. When the organization intends to automate these processes, it has to plan accordingly. The automation may be done by the built in information technology division or the job may be outsourced. When it is outsourced, the company which is going to build the information system has to plan the project. It will do this task by finding various facts in the organization (i.e. tracking the flow of information in the absence of automation as well as obtaining clarity about various processes). Ultimately, it has to prepare the schedule for the job and start gathering requirements. After requirements are gathered and a consensus is reached on the requirements that are to be considered for the information system, a modular and structured design for the system has to be explored and ultimately, system has to be designed.

This block consists of 3 units and is organized as follows:

Unit-5 focuses on various fact finding techniques such as Interviews, Group Discussions, Presentations etc. Also, various issues involved in the study of feasibility of developing such a system are discussed. The process of doing a cost benefit analysis is explained and various requirements gathering techniques are discussed.

Unit-6 deals with various design principles such as Top down design and Bottom up design. Also, various symbols that are used in the Structure charts are indicated and explained. The concepts of Cohesion and Coupling are explained in this unit.

Unit-7 deals with the design of information system. Various modeling techniques such as Data Flow Diagrams and E-R diagrams are explained. Also, various process specification tools such as Decision Tables, Decision Trees and Structured English notation are also explained with examples in this unit.

UNIT 5   PROCESS OF SYSTEMS PLANNING

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 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.

Security

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

Services

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

Efficiency

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

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

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:
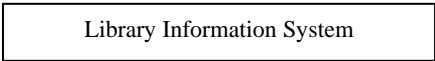
Library Information System

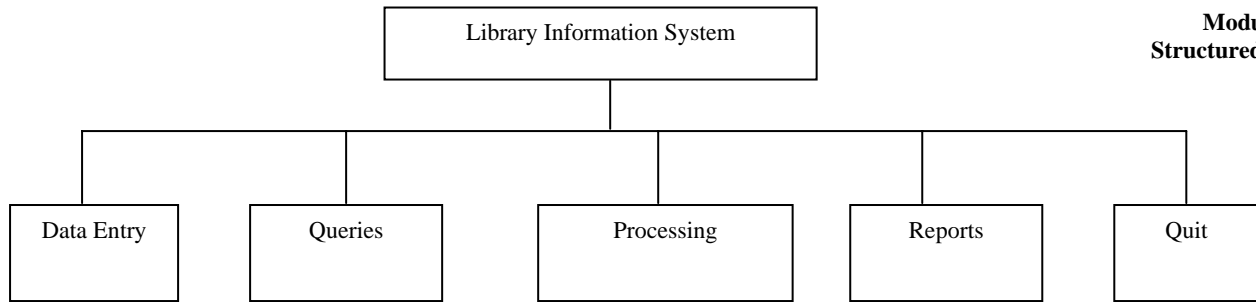**Figure 6.1: The top (root) of software system**

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.
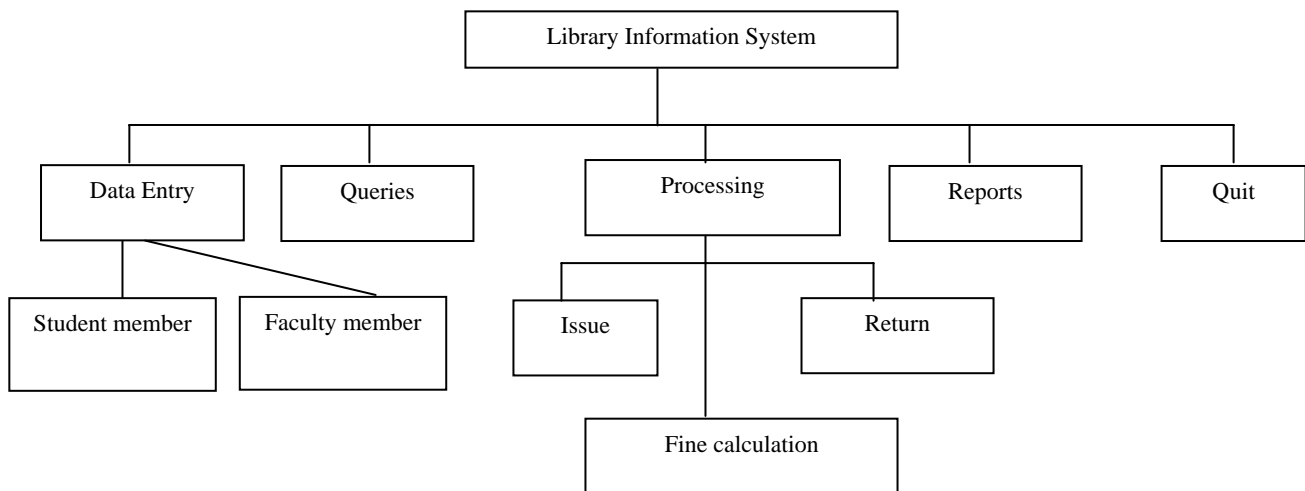

Figure 6.3: Hierarchy Chart of Library Information System

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.
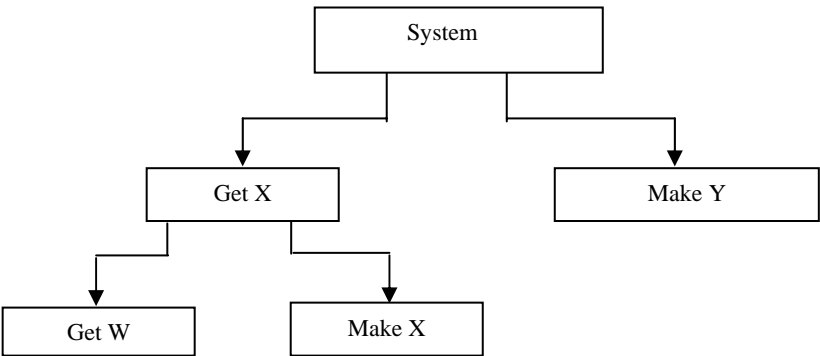

Figure 6.4: Hierarchy of a Structure Chart

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.
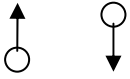
Figure 6.5: Data couples   **Figure 6.6: Control Flag**

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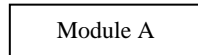
Module A

Figure 6.7: Symbol for Module

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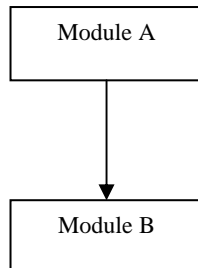
Module A

Module B

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.
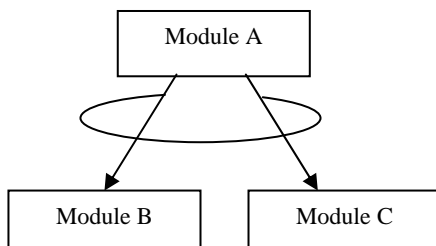
Module A

Module B   Module C

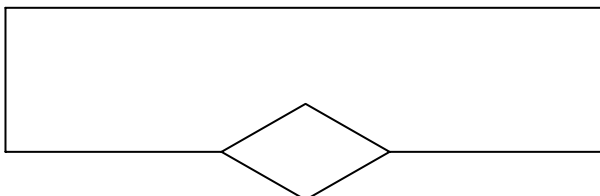**Figure 6.9: Repeated calls of Modules B and C by Module A**

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.
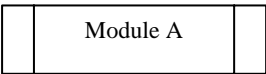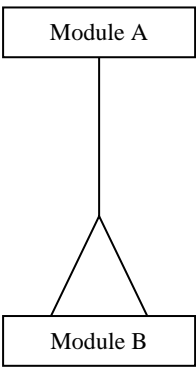

Figure 6.11: Predefined module A

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.
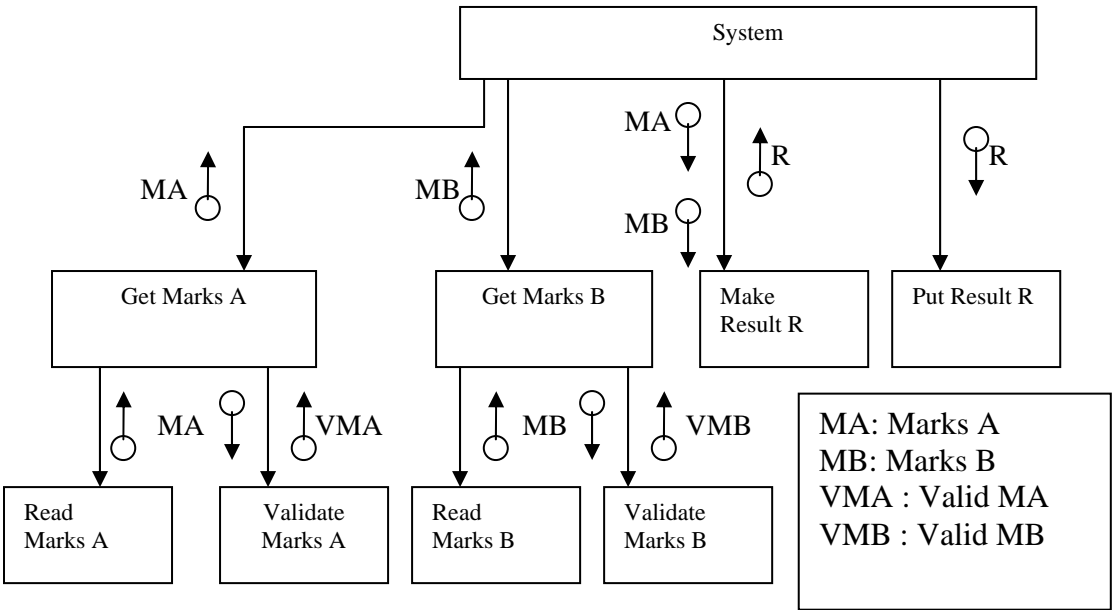

Figure 6.13: Reading a Structure chart

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 of range of size can be defined on modules though it is done occasionally. The size varies from module to module and from project to project.
5. A module should not be assigned the duty of performing more than one function.
6. The coding of modules should be generic. It enables the system to use the module as frequently as possible.

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

- A system should be divided into as many relatively independent modules as possible. This is known as *factoring*.
- A superordinate module should control not more than seven subordinate modules. Of course, this guideline is not strict and varies from system to system.
- The dependency levels between modules should be minimum. This automatically leads to the design of modules, which don't communicate, frequently with each other. Also, the communication between modules should be through parameters. Of course, Boolean variables or flags can be used for the purpose of communication. This is called *coupling*.
- Usually, a module if of not more than 100 lines. It may be a minimum of 50 lines. But, these sizes are not to be strictly followed and they may vary from system to system. It is notable here that lesser 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.
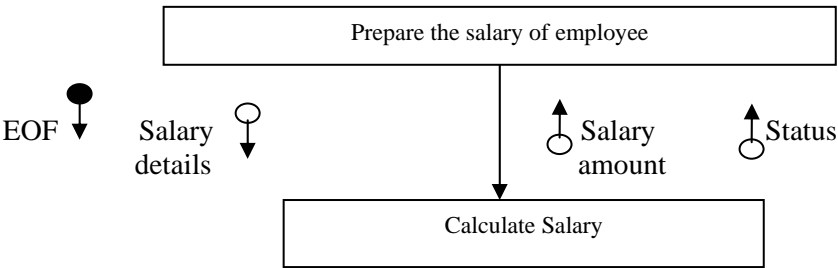

Figure 6.14: Example of Data Coupling

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.
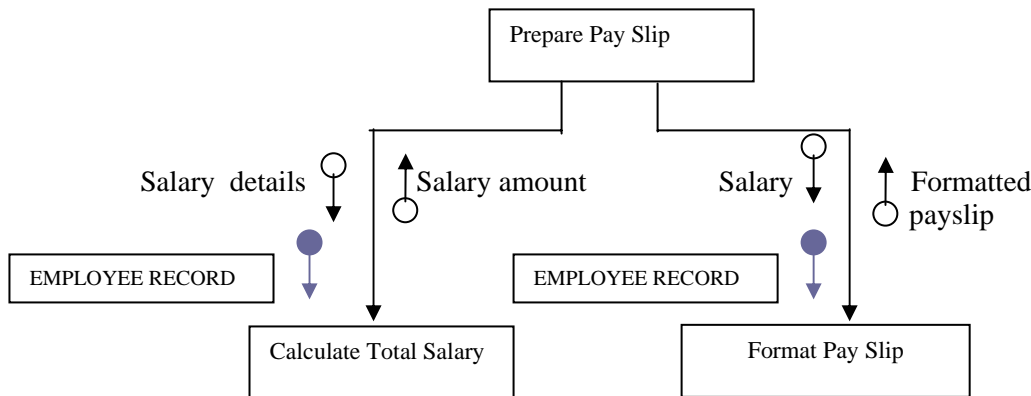

Figure 6.15: Example of Stamp Coupling

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.
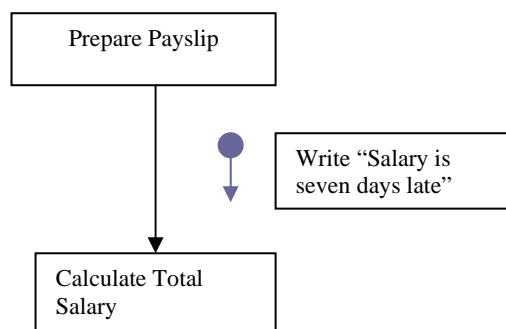

Figure 6.16: Example of Control Coupling

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.
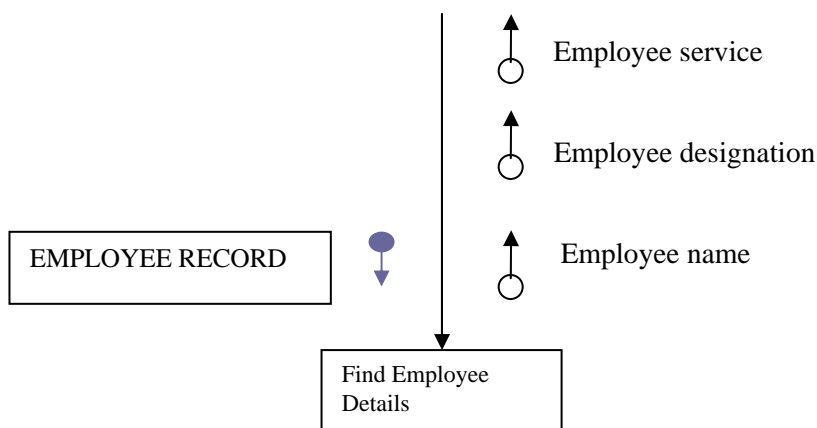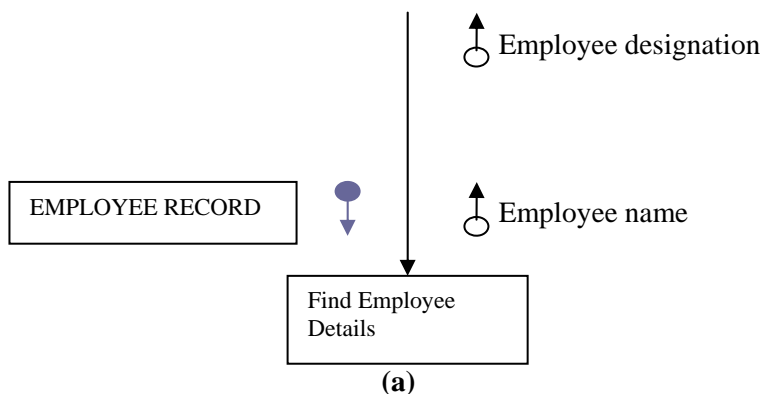
Employee service

Employee designation

EMPLOYEE RECORD

Employee name

Find Employee
Details

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.
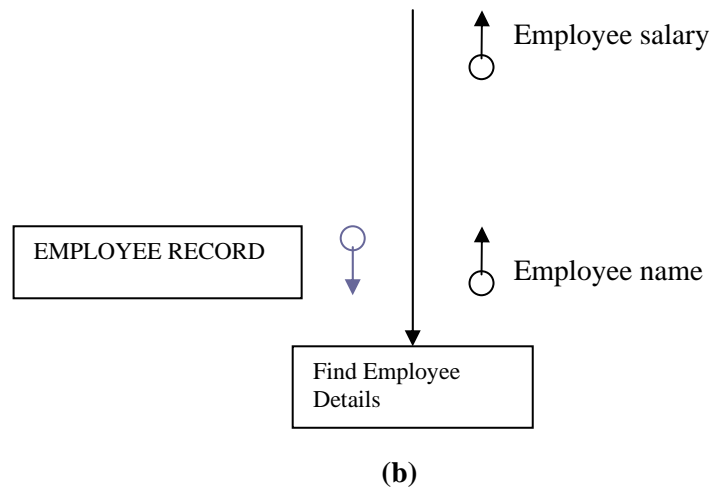
Employee designation

EMPLOYEE RECORD

Employee name

Find Employee
Details

(a)

Employee salary

EMPLOYEE RECORD

Employee name

Find Employee Details

(b)

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 studies 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

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.
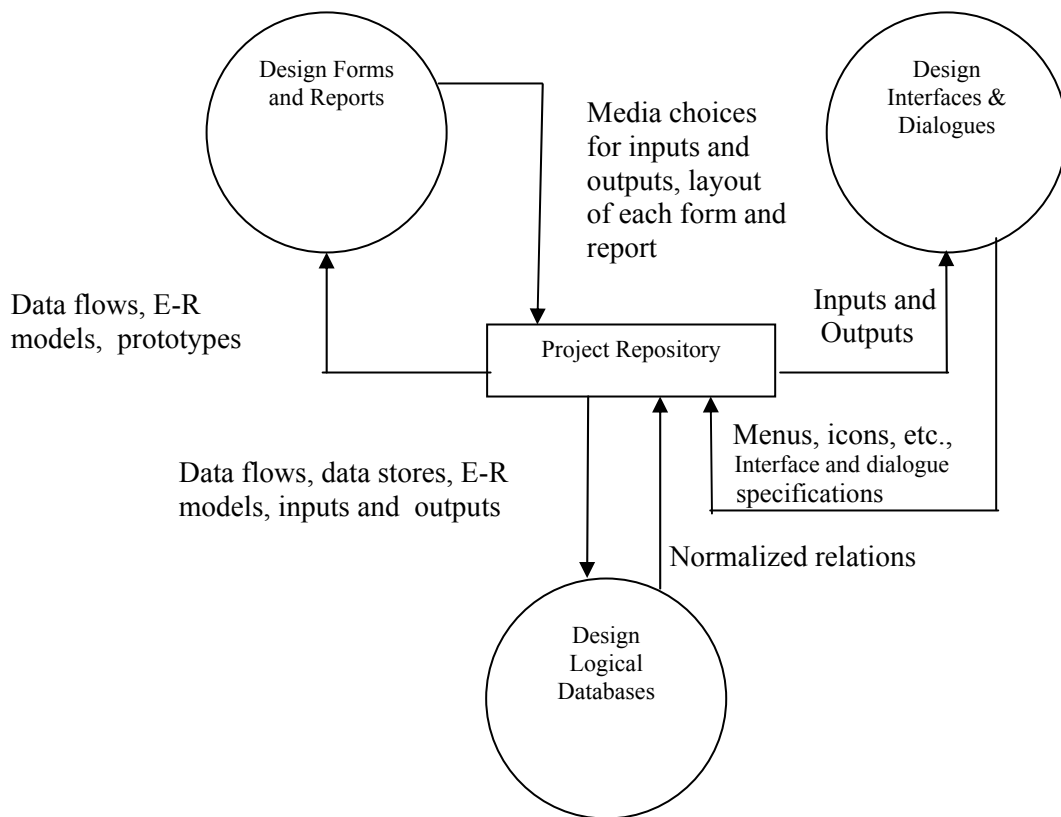

Figure 7.1: Steps in Logical Design

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.

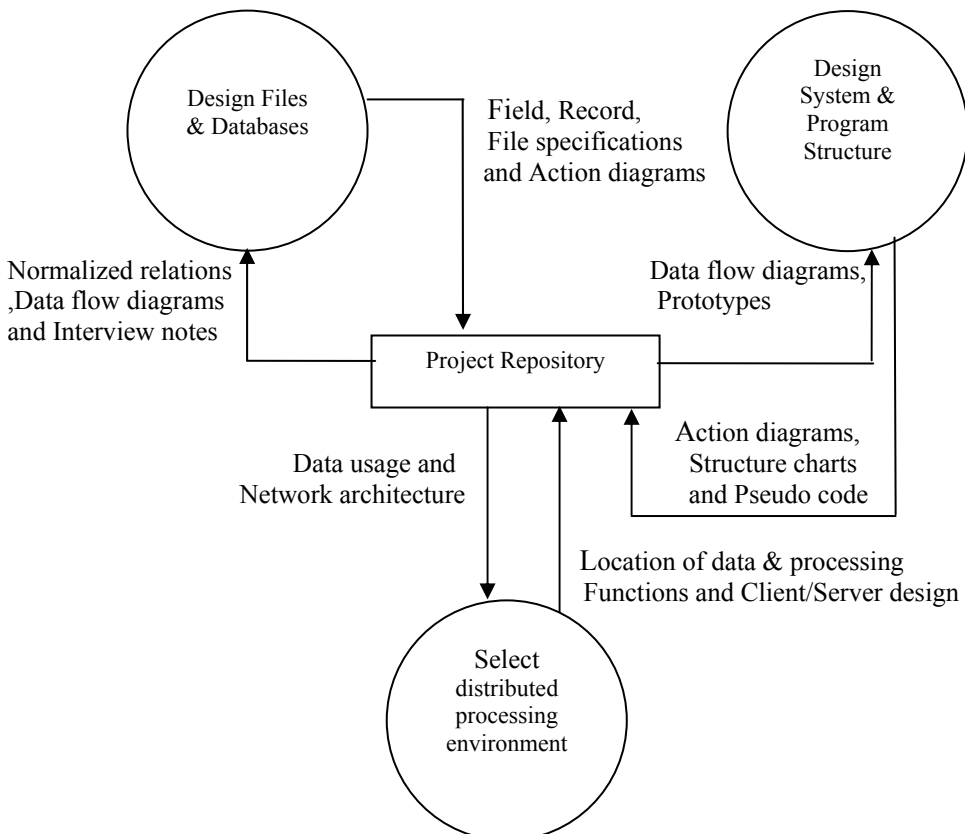Figure 7.2 depicts various steps involved in physical design.

Figure 7.2: Steps in Physical Design

35

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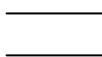
The notation (given by Yourdon) for source or sink is

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:
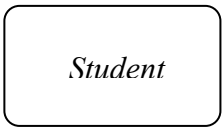
Student

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.
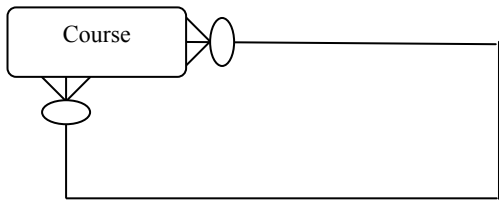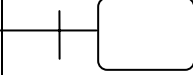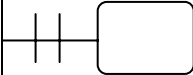
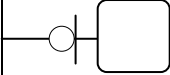**Figure 7.3: Example of Recursive relationship**
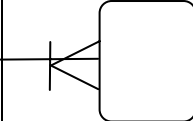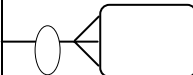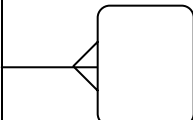
Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one (One and only one)	1	1	or
Zero or one	0	1	
One or more	1	Many (>1)	
Zero, one or more	0	Many (>1)	
More than one	>1	>1	

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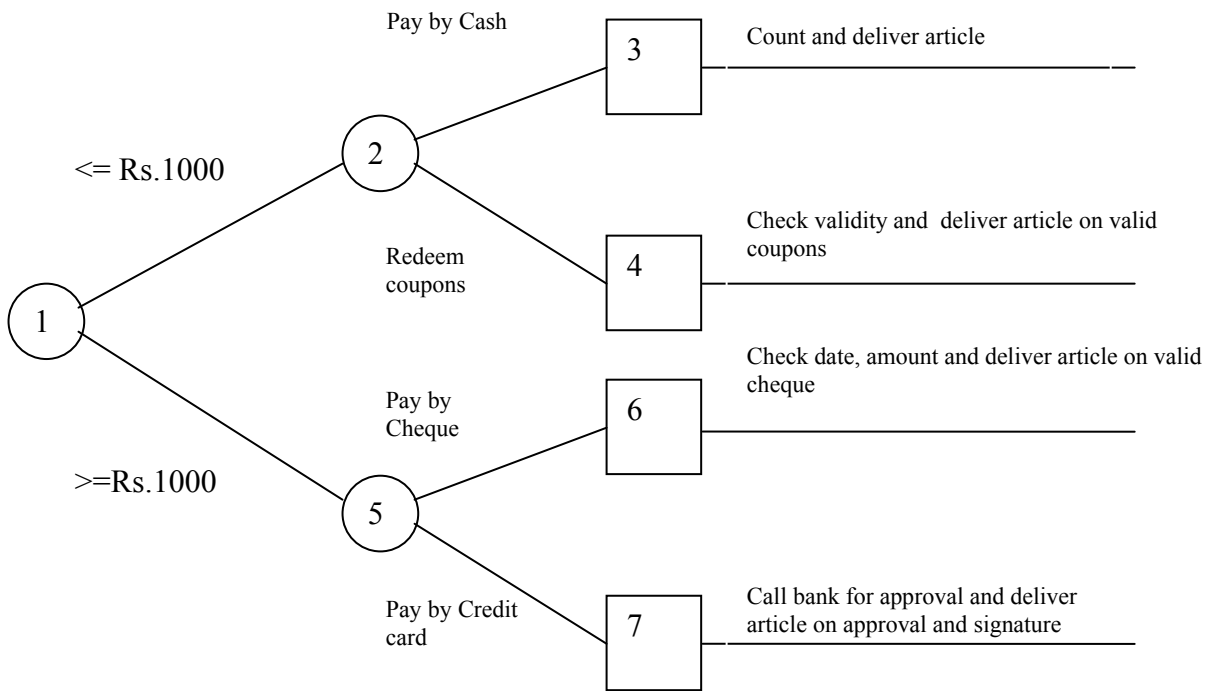
Figure 7.6: An Example Decision tree for a Customer Bill Payment System

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
Substract (-)	Greater Than (>)
Exponentiate (**)	Less Than (<)
	Less Than or Equal To (<=)
	Greater Than or Equal to (>=)
	Equals to (=)
	Not equal to ($\neq$)

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 withdrawls. 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 with out 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:
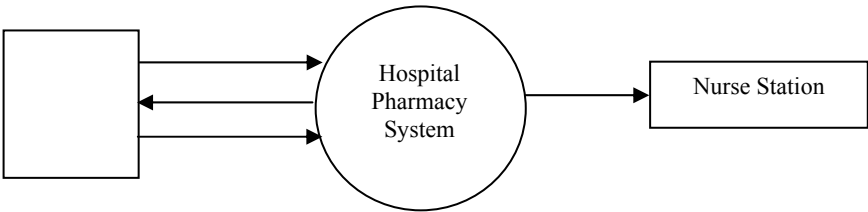
The following is the first level DFD:

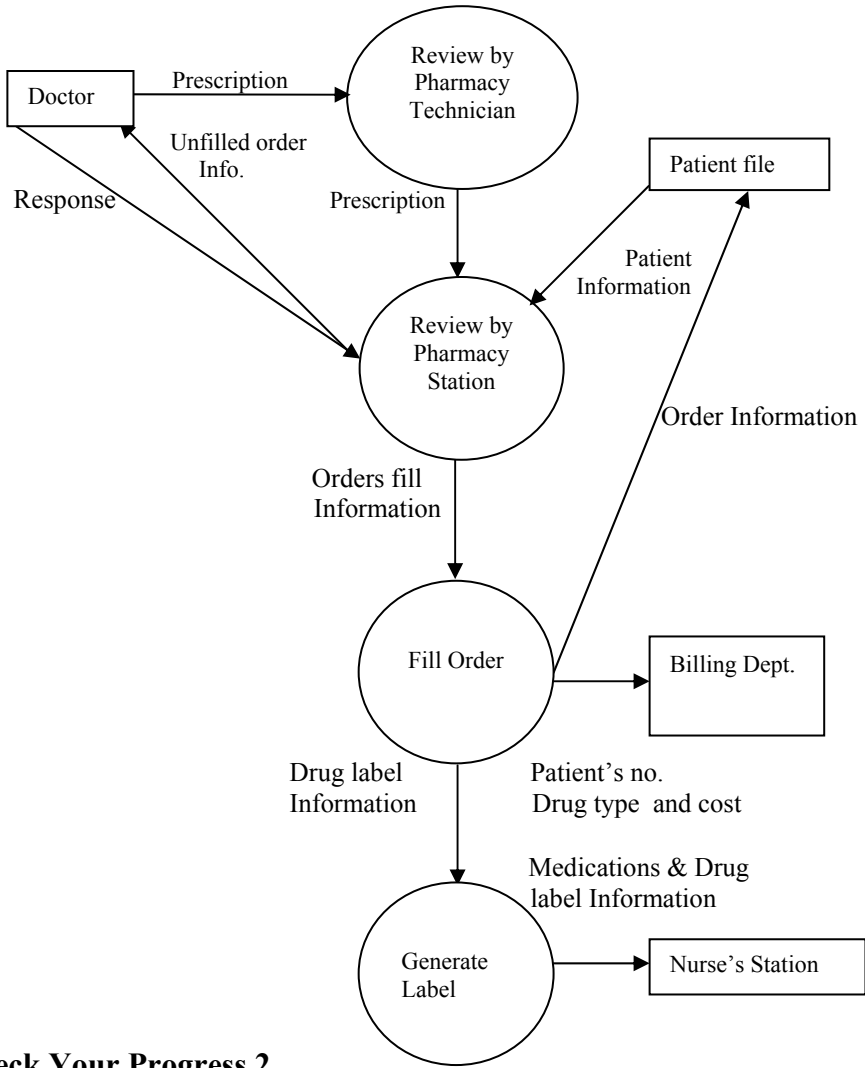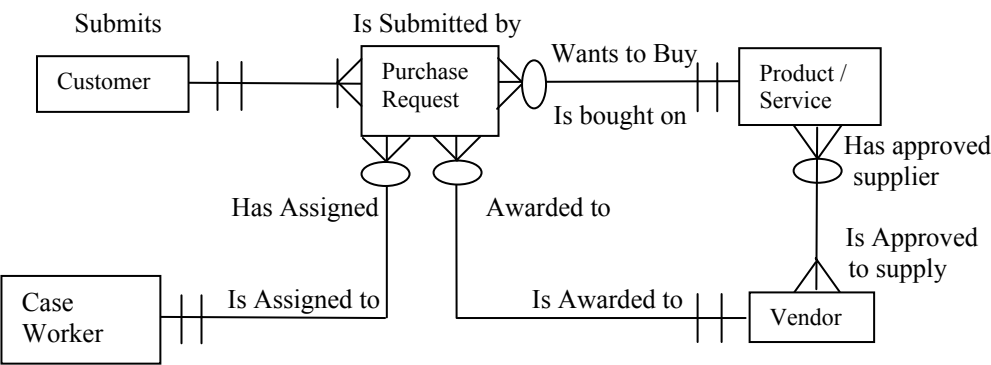## Check Your Progress 2

1. The following is the ER diagram:

Check Your Progress 3

1. We have to identify conditions and values.

Data Elements Condition	Values
1. Account type	R = Regular
	S = split
2. Insurance	Y = Yes
	N = No

45

3. Balance Dropped below Rs.2000/- during month?   Y = Yes
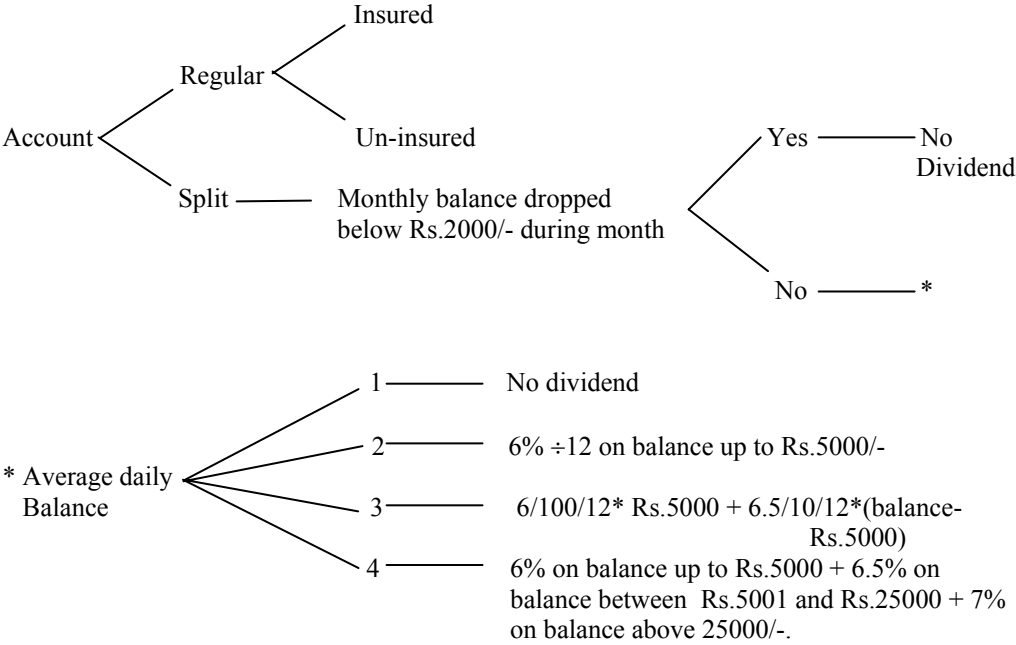
Steps for Decision Table

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:

        YN YN YN YN

    b. For condition two,

         size of pattern that repeats in step (a) i.e. 2

        => YY NN YYNN

    c. For condition three,

        size of pattern that repeats in step (b) i.e., 4 so the possible combination is:

        YYYY NNNN

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.

```
                         Insured
            Regular  <
 Account  <              Un-insured                 Yes ——————— No
                                                                Dividend
            Split ——————  Monthly balance dropped  <
                          below Rs.2000/- during month
                                                    No ——————— *
```

```
                    1 —————— No dividend

                    2 —————— 6% ÷12 on balance up to Rs.5000/-

 * Average daily  <
   Balance          3 —————— 6/100/12* Rs.5000 + 6.5/10/12*(balance-
                                                            Rs.5000)
                    4 —————— 6% on balance up to Rs.5000 + 6.5% on
                             balance between  Rs.5001 and Rs.25000 + 7%
                             on balance above 25000/-.
```

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