Information Systems Analysis and Design

2nClass

2024-2025

فرع نظم المعلومات

الدكتور: أحمد عبد الزهرة شكارة

General Introduction

Information systems are crucial to the success of modern business organizations, and new systems are constantly being developed to make businesses more competitive. The key to successful system development is through systems analysis and design to understand what the business requires from the information system.
System analysis and design is used to analyze, design and implement improvements in the functioning of businesses that can be accomplished through the use of computerized information systems.

What is Information Systems Analysis and Design?

- A method used by companies to create and maintain systems that perform basic business functions
- Main goal is to improve employee efficiency by applying software solutions to key business tasks
- A structured approach must be used in order to ensure success

Systems and Information Systems Concept

Data:

Are raw facts or elementary descriptions of things, events, activities, and transactions that are captured, recorded, stored, and classified but not organized to convey any specific meaning. It can be in the form of numbers, text, images, or any other type of input.

Information:

It is a collection of raw facts (data) that has been processed, analyzed and organized in a meaningful way to become useful. for example, if we include the student's name with his or her grades, customer names with bank balances, and employees' wages with hours worked, we would have useful information.

Knowledge:

is the use of human experience with data and information for the purpose of deriving a set of rules that helps to make decisions. In fact, we cannot store knowledge because it depends on human theoretical and practical experience to the subject.

Note:

Information that is not of high quality may lead to wrong decisions and cost the organization a great deal of money. Therefore, for information to be useful to managers and the organization, the information must contain characteristics such as being accurate, complete, reliable and flexible.

System:

Is an organized collection of people, documents, data, procedures, machine, or any other entities such that they interact with each other as well as with the

environment to reach a predefined goal."
So, all systems have some common properties, they have:

- Elements.
- Environment.
- Interaction between elements and the environment.
- Goals to be fulfilled.

Note: We use the term "Information" as a general term, without differentiating the meaning.

General Model of a system

A general model of a system is input, process, and output. The system is inside the boundary; the environment is outside the boundary.

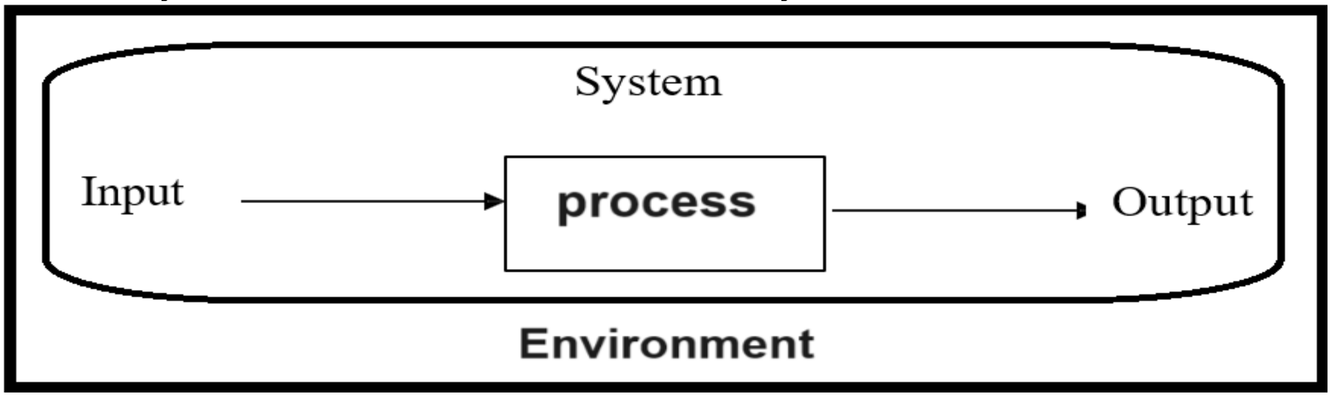


Figure 1: General model of a system

Each system is composed of subsystems, which in Turn are made up of other subsystems, each subsystem being delineated by its boundaries. The interconnections and interactions between the subsystems are termed *interfaces*. Interfaces occur at the boundary and take the form of inputs and outputs.

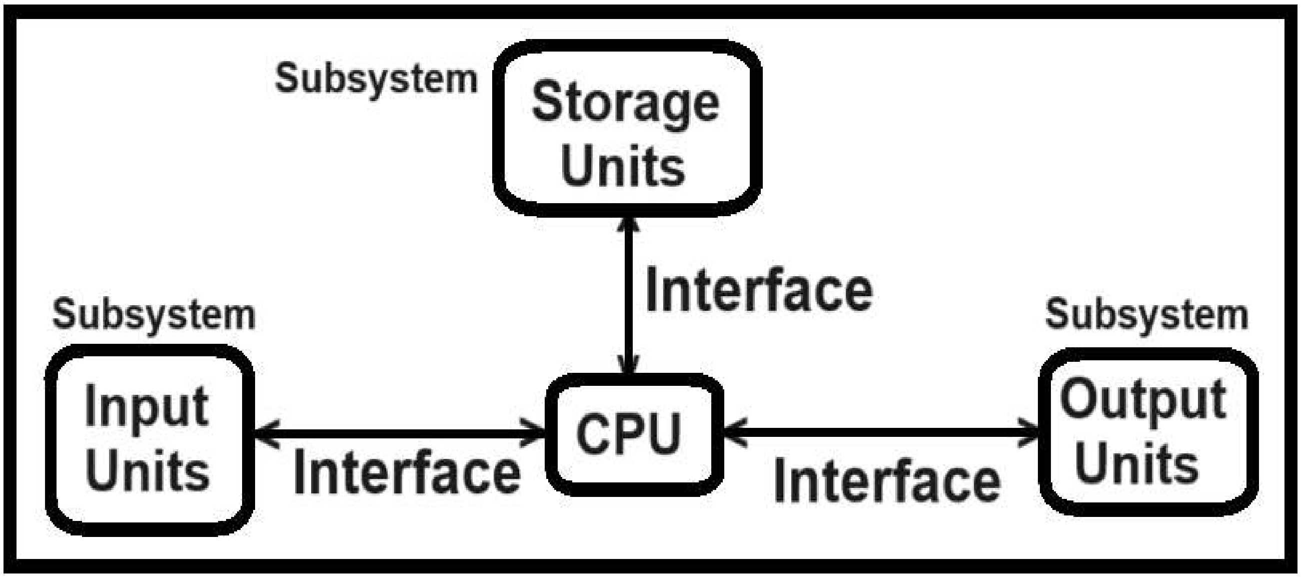


Figure 2: Computer System

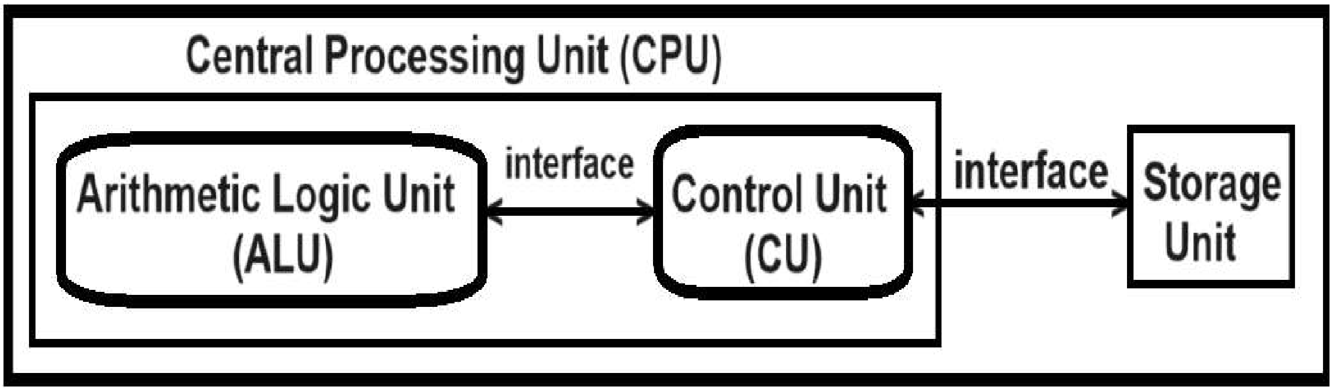


Figure 3: Central Processing Unit System

Types of Systems

There are several ways of classifying systems that is closed and open systems or deterministic and probabilistic.

Closed and Open Systems

Closed system

A closed system is defined in physics as a system, which is self-contained. It does not exchange material, information, or energy with its environment.

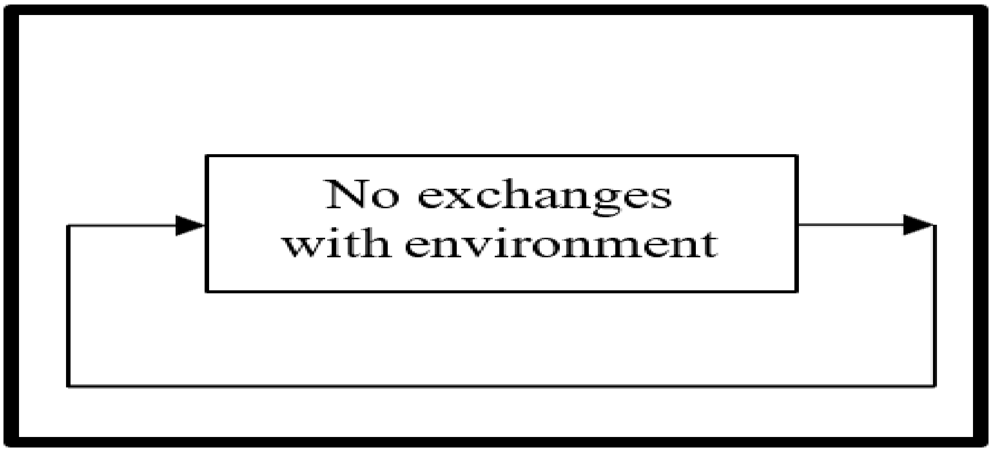


Figure 4: Close System

Relatively closed system

A relatively closed system is one that is not subject to external disturbances and has specific, controlled inputs and outputs. A computer program is a relatively closed system because it accepts only previously defined inputs, processes them, and provides previously defined outputs.

Open system

Open systems exchange information the material or energy with the environment, including random and undefined inputs. Open systems tend to have form and structure to allow them to adapt to changes in their environment in such a way as to continue their existence. They are "self-organizing" in the sense that they change their organization to response to changing conditions.

Deterministic and Probabilistic Systems

Deterministic Systems

A deterministic system operates in a predictable manner. The interaction among the parts is known with certainty. If one has a description of the state of the system at a given point in time plus a description of its operation, the next state of the system may be given exactly, without error. An example is a correct computer program, which performs exactly according to a set of instructions.

Probabilistic Systems

A probabilistic system can be described in terms of likely behavior, but there is always some degree of error associated with predicting what the system will do. An example of a probabilistic system is a weather system that may predict rainfall but its predictions may not come true for a variety of reasons.

Information Systems and Organization

Information system:

Is a system that serves to provide information within the organization when and where it is needed at any managerial level. Such a system must take the information received and store, retrieve, transform, process, and communicate it using the computer system or some other means.

The Information must have the following requirements:

- Its recipient in the proper frame of reference must understand it.
- It must be relevant to a current need in the decision-making process.
- It must have a surprise value, that is, what is already known should not be presented.
- It must lead its users to make a decision.

Information system to yield information with above characteristics, it must have some attributes, as follows:

- Effective processing of information. This refers to proper editing of input data and efficient utilization of hardware and software.
- Effective management of information. Stressed are care in file management and security and integrity of existing data.
- Flexibility. The information system should be flexible enough to handle a variety of operations.
- User satisfaction. Of prime importance are user understanding of satisfaction with the information system.

Any action in an organization depends on the result of a decision-making process at the proper managerial level of that organization.

Subsystems of an Information System

The information system of any organization contains information related to three basic types of operations, namely, transaction processing, control, and strategic planning. One could group them into two as operating level and management level activities as,

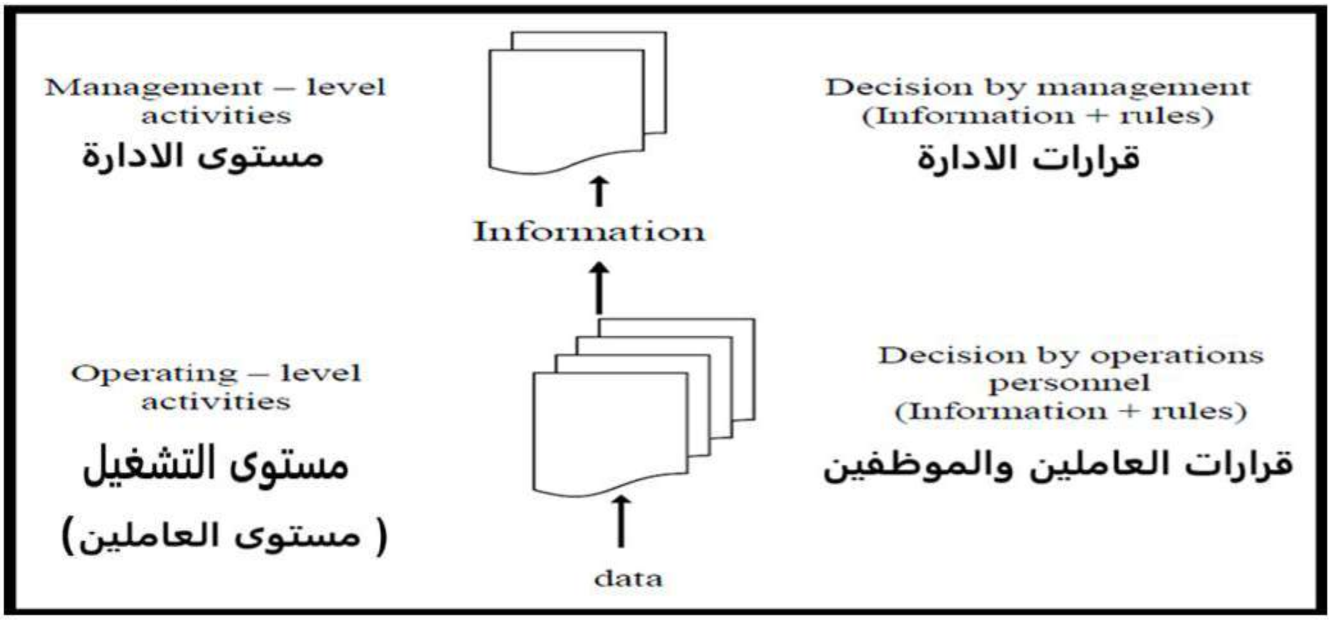

Management – level activities
مستوى الادارة
Decision by management (Information + rules)
قرارات الادارة
Information
Operating – level activities
مستوى التشغيل
(مستوى العاملين)
Decision by operations personnel (Information + rules)
قرارات العاملين والموظفين
data

Figure 5: Operating level and Management level

Operating level and management level activities

This figure represent the management level activities as a triangle and to base it on a rectangle as a symbol of operating level activities.

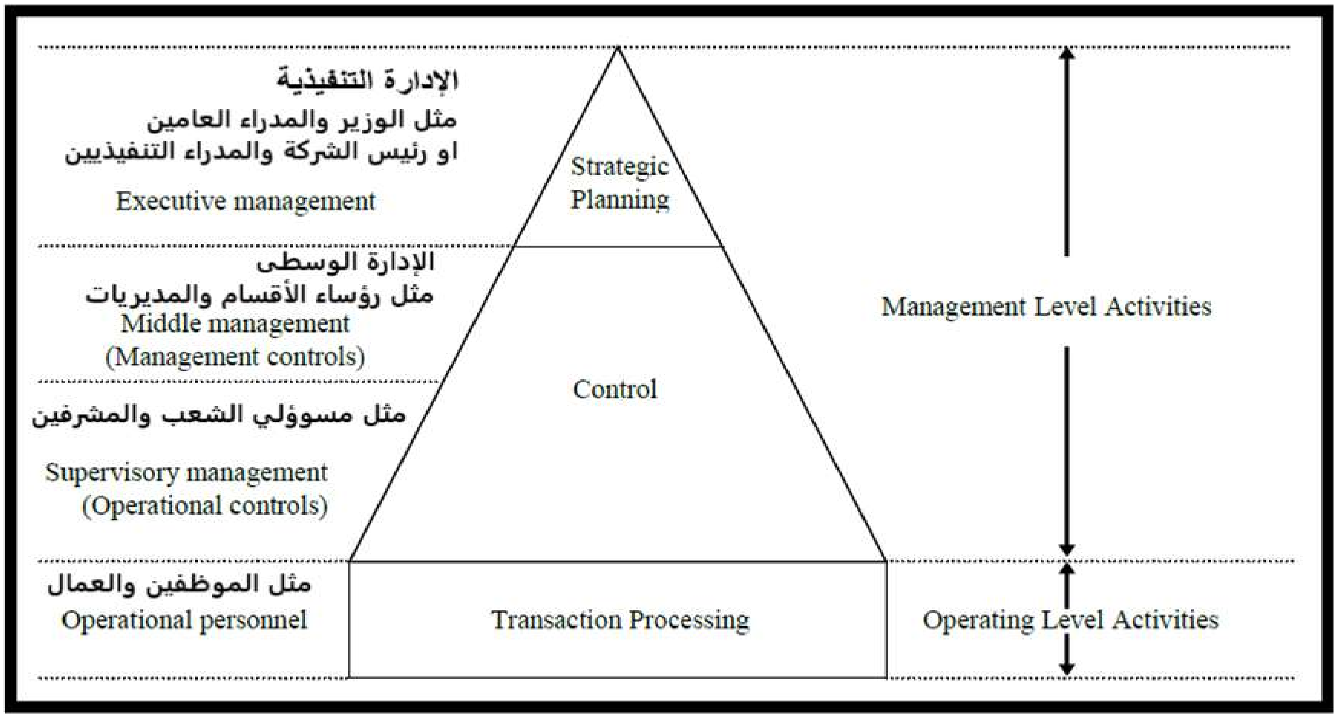


Figure 6: Information – related activities of organization

Strategic planning

In strategic planning, the executive or top management of the organization decides on the objectives of the organization, on the resources to be used to attain these objectives. The control function has both management and operational components. In management control, middle-level managers assure those resources are obtained and used effectively and efficiently to accomplish the organization's objectives. In operational control, supervisory management assures that specific tasks are carried out effectively and efficiently.

Transaction processing

It refers to the daily routine business operations of the organization and the relationship between the organization's information system and its activities. The organization's information system consists of two main subsystems:

- **Management Information System (MIS)**

  is the information subsystem relevant to managerial decision for control and strategic planning purposes. In addition, the tendency now is to break down the MIS subsystem of an information system into MIS and DSS (Decision Support System).
- **Operations Information System (OIS)**

  It is an information subsystem that deals with processing transactions in an organization. It is often referred to as electronic data processing (EDP) if computers support the flow of information. The internal output of an EDP system consists mainly of factual reports and summary reports.

EDP/ MIS/ DSS

In a MIS, data from a variety of potential users is transformed into meaningful information that is used primarily by middle and upper management. In contrast to the EDP system, an MIS views not just the transformation of data but how it can be turned into useful organizational information. The major output of a MIS consists of standardized reports and interrogative reporting. A MIS is designed from an organizational perspective rather than from business transaction perspective.

In DSS, the emphasis is on decision making at all levels of management in the organization. DSS is a tool to help managers in their decision-making activities by providing them with the necessary information. The information produced by a DSS consists of interactive/iterative reports and unstructured reports oriented to the individual manager.

An important distinction between DSS on the one hand and EDP and MIS on the other has to do with development of the system structure. In EDP and MIS, the system designers develop the system structure, whereas in DSS the manager not only provides the input but he/she defines the structure as well.

Data communications system

The data communications system of an organization is another topic that is closely related to the information system of that organization. The transmission of data / information from one point to another in the organization by any means is data communications, and it seems that more and more, data communication and information systems of organizations are becoming inseparable.

One of the major advantages of dB applications is data integrity or data correctness. A dB also improves the value of information by minimizing data redundancy. Another major advantage of dB is that the same set of data is available for different applications.

Another important factor related to the success of information systems as they impact decision-making activities is the reliability of the information in the database system.

Black-box System

A subsystem at the lowest level (input, process, and output) is often not defined as to the process. This system is termed a *black box*, since the inputs and outputs are known but not the actual transformation from one to the other.

Systems Analysis

These are some principles that help analyze old systems (for example, paper-based systems) to build modern, advanced systems (automated computer systems).

Decomposition

A complex system is difficult to comprehend when considered as a whole. Therefore, the system is decomposed or factored into subsystems. The boundaries and interfaces are defined, so that the sum of the subsystems constitutes the entire system. This process of decomposition (or process of factoring) is continued with subsystems divided into smaller subsystems until the smallest subsystems are of manageable size.

Decomposition into subsystems is used both to analyze an existing system and to design and implement a new system. In both cases, the investigator or designer must decide how to factor, i.e., where to draw boundaries. The decisions will depend on the objectives of the decomposition and also on individual differences among designers. In design the boundary needs to be clearly specified, interfaces simplified, and appropriate connections established among the subsystems.

Simplification

The process of factoring could lead to large number of subsystem interfaces to define. Simplification is the process of organizing subsystems so as to reduce the number of interconnections. One of the common use methods of simplification is *Clusters* of subsystems are established which interact with each other, then a single interface path is defined from the cluster to other subsystems or clusters of subsystems. An example is a database, which is accessed by many programs, but the interconnections only through a database management interface.

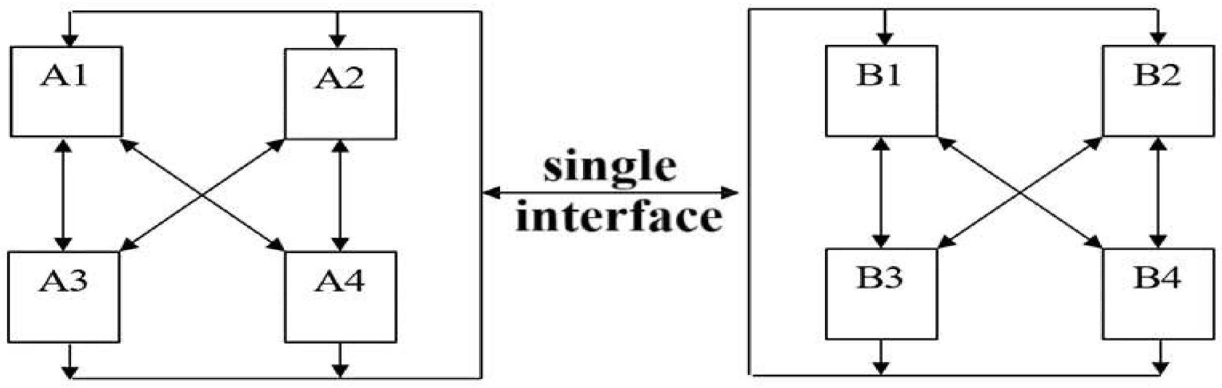


Figure 7: Systems connected within cluster, and clusters connected with single interface

Feedback control

The basic model of the system usually contains inputs, processing operations and outputs, and does not include feedback. In order to improve the performance of the system, the feedback process is added later and a feedback control subsystem is added.

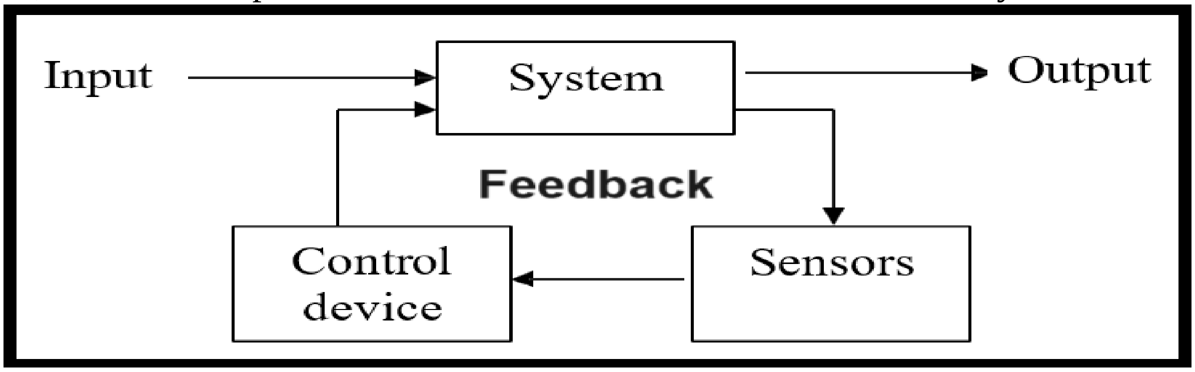


Figure 8: Feedback control for a system

Positive and Negative feedback

Feedback, which seeks to dampen and reduce fluctuations around the standard, is termed negative feedback. It is used in feedback control loops. Positive feedback reinforces the direction in which the system is moving. In other words, positive feedback causes the system to repeat or amplify an adjustment or action.

Negative Feedback Control

Negative feedback control in a system means keeping the system operating within certain limits of performance. Control using negative feedback normally involves four elements:

1- A characteristic or condition to be controlled; The characteristic or condition must be measurable from some output.
2- A sensor for measuring the characteristic or condition.
3- A control unit which compares the measurements with a standard for that characteristic or condition.
4- An activating unit, which generates a corrective, input signal to the process.

These elements are diagrammed as:

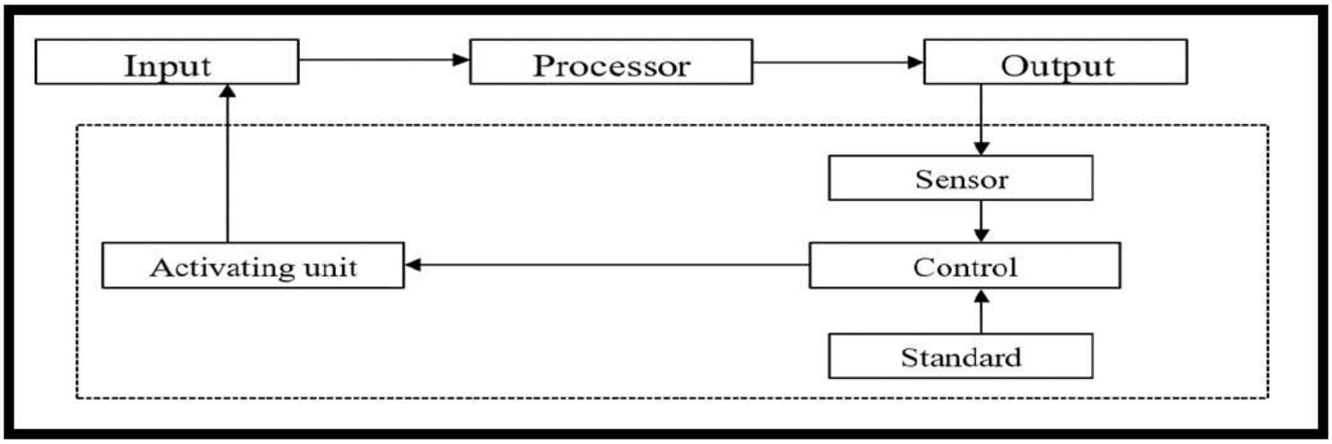


Figure 8: Negative Feedback Control Elements

Feedback Control Loops

Feedback control loops are frequently classified as closed or open.

A closed control loop

An automated control such as a thermostat or computer-controlled process. In much the same way that a closed system in insulated from disturbances in the environment, a closed feedback loop is insulated from disturbances in the control loop.

An open control loop

It is one with random disturbances, such as those associated with human control elements, there are variations between the two extremes.

Filtering

Filters are often used for system input and in feedback. A filter is essentially a system element, which keeps out certain inputs, allowing others to enter the system. Filters may be used to:

1. Reduce the type of inputs.
2. Reduce the amount of information.

In summary filters should be used in information systems to reduce unneeded or irrelevant data being accepted for processing or being output for another (perhaps human) subsystem.

System Life Cycle

System Life Cycle

Computer-based information systems are developed in a sequence of steps. This sequence is known as the system life cycle or problem-solving cycle or sometimes the development cycle. There are several types of life cycles, each used for a specific purpose, but the most common type is the linear life cycle.

linear life cycle

The linear cycle is made up of a sequence of consecutive phases. No phase in the sequence can commence until the previous phase has been completed. A report is produced at the end of each linear cycle phase, describing what has been achieved in the phase and outlining a plan for the next phase. The report also includes any system descriptions, design decisions and problems encountered. Analysts and designers use this information at the next phase. Phase reports are also used to keep management informed of project progress, so that the management can use the reports to change project direction and to allocate resources to the project.

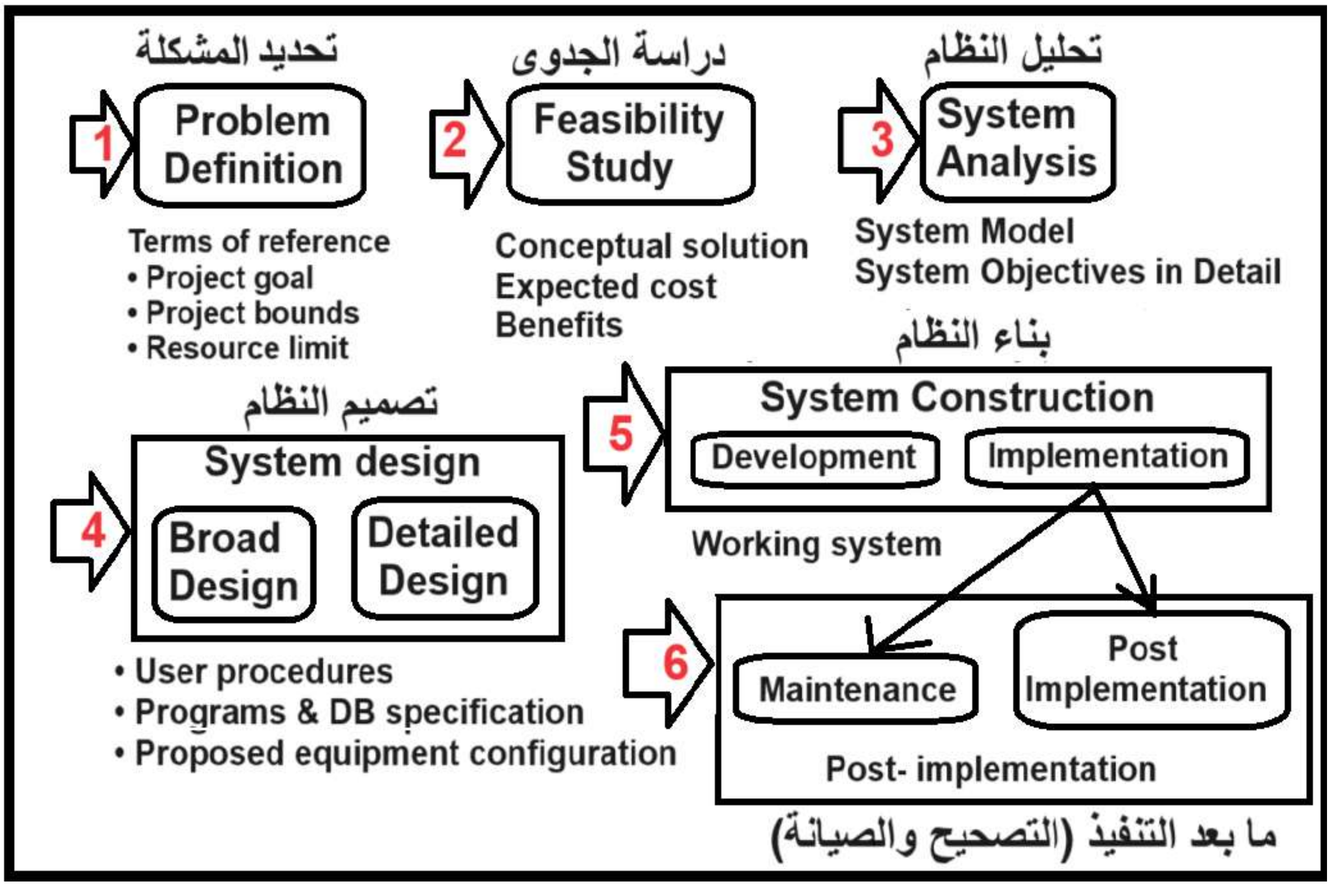


Figure 9: The linear cycle

Linear Cycle Phases

Phase 1 – Problem definition

It defines the problem to be solved and It also defines the project bounds, what parts of the system can be changed by the project and what parts are outside its control. The resources to be made available to the project are also specified in this phase. These three important factors – the project goal, project bounds, and resource limits – are sometimes called the project's terms of reference. Because of their importance, the organization's management sets them.

Phase 2 – Feasibility Study

The feasibility study proposes one or more conceptual solutions to the problem set for the project. The conceptual solutions give an idea of what the new system will look like. They define what will be done on the computer and what will remain manual. They also indicate what input the systems and what outputs will be needed will produce. These solutions must be feasible and a preferred solution must be accepted.

Three things must be done to establish feasibility:

- First, it is necessary to check that the project is technically feasible; does the organization in fact have the technology and skills necessary to carry out the project and, if not, how should the required technology and skills be obtained?
- Second, operational feasibility must be established; to do this it is necessary to consult the system user to see if the proposed solution satisfies user objectives and can be fitted into current system operations.
- Third, project economic feasibility needs to be checked. The study must determine whether the project's goals can be achieved within the resource limits allocated to it. It must also determine whether it is worthwhile to proceed with the project at all or whether the benefits obtained from the new system are not worth the costs (in which case, the project will be terminated).

Phase 3 – System Analysis

A detailed analysis of the system is made in this phase to familiarize designers with the operation of the existing system. analysts use many of the commonly used system analysis techniques such as data flow diagrams and data analysis during their analysis. They also follow specified procedures in their search for information about the system. this calls for interviews with system users, questionnaires and other data-gathering methods. Analysts must spend considerable time examining components, such as the various forms used in the system, as well as the operation of existing systems.

Phase 4 – System Design

This phase produces a design for the new system. There are many things to be done here:

- Designers must select the equipment needed to implement the system.
- They must specify new programs or changes to existing programs.
- Specify a new database or changes to the existing database.
- Designers must also produce detailed user procedures that describe how users will use the system.

System design usually proceeds in two steps.

Phase 4A – Broad design.

During Broad design, the following is done:

- Determine which parts of the system will be automated and which parts will remain manual.
- Propose new key functions and identify changes to existing functions.
- Identify the important inputs and outputs at this stage.
- The proposed conceptual solutions are considered in more detail through the feasibility study.

It is common during broad design to consider a number of alternative solutions, involving different degrees of automation. One of these alternatives is then chosen as the preferred broad solution.

Phase 4B – Detailed design.

It is only when abroad solution is chosen that detailed design starts. During the detailed design phase, the database and program modules are designed and detailed user procedures are documented. The interfaces between the system users and computers are also defined. These interfaces define exactly what the user will be expected to do to use the system.

Phase output

The output of this phase includes a proposed equipment configuration together with specifications for the database and computer programs. Detailed user procedures are also provided. These include any input forms and computer interfaces between users and the computer. The user manual is also ready at the end of this phase.

Phase 5 – System Construction

This phase is up into two smaller phases:

phase 5A – Development

During development, the individual system components are built. Programs are written and tested, and user interfaces developed and tried by users. The database is initialized with data.

Phase 5B – Implementation.

During implementation, the components build during development are put into operational use. To complete the changeover, users must be trained in system operation and any existing procedures converted to the new system.

Phase output

At the end of the phase, users are provided with a working system. This includes the set of working programs and an initialized database. In addition, any system documentation describing the programs is also completed. All users have by now been trained and can use the new system.

What happens after project completion?

The system is considered to be working when Phase 5 has been completed. However, there are still a number of activities that take place after a system is completed. The two main activities are the *post-implementation review* and *maintenance.*

Maintenance

Maintenance is necessary to eliminate errors in the working system during its working life and to tune the system to any variations in its working environment. There always some errors detected that must be corrected.

Post-implementation review

The *post-implementation review* usually takes place about a year after the system is implemented. It evaluates the new system to see if it has indeed satisfied the goals set for it. The system is examined to see if the benefit expected of it have been realized. If they have not, a study is made to see why not. Part of this study is the project life cycle itself. This evaluation then sets guidelines for decision making in future projects. In addition, post-evaluation may suggest minor changes to be made to the system. if the system is performing badly, post evaluation may suggest a total redesign.

Staged Design

We will not that each stage starts with its own problem definition and feasibility study. This is necessary to get a detailed evaluation and plan for each stage, and also to evaluate the feasibility of integrating the system produced in each stage with the systems developed in previous stages. The stage then continues with systems analysis, system design, and system implementation.

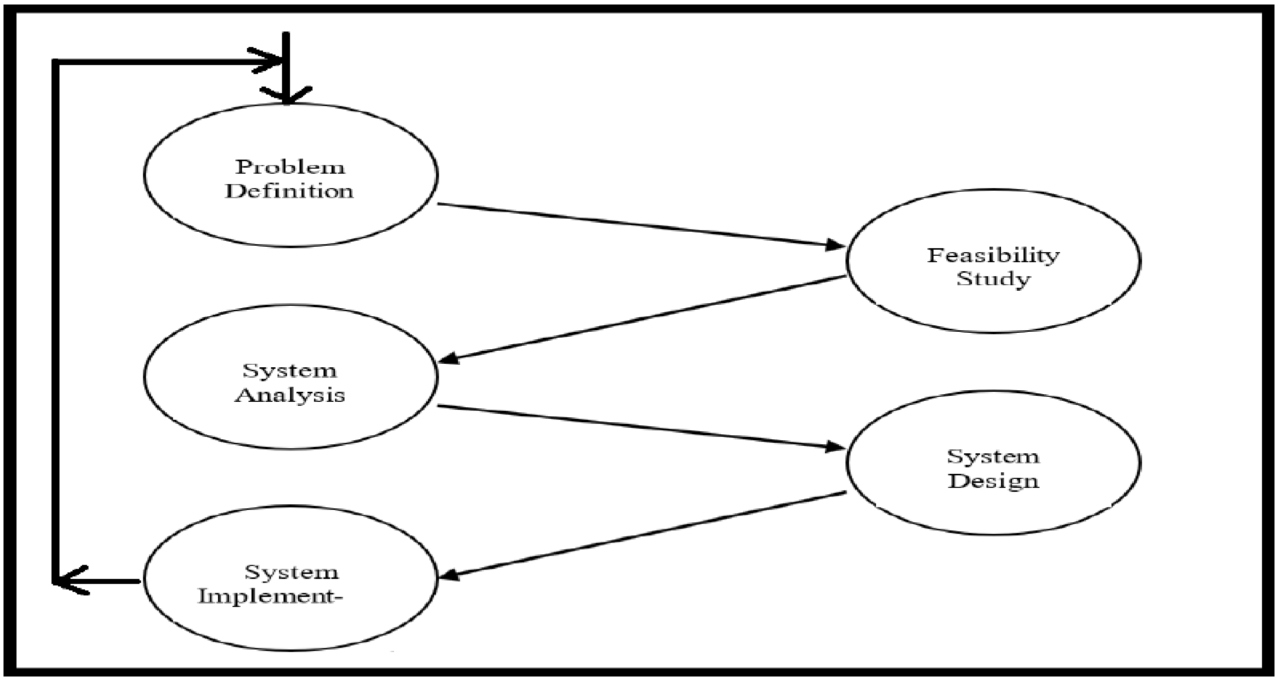


Figure 10: Staged Design

Problem solving cycle are often called loopy linear. They arise by default because something was overlooked in the original plan. Iteration, or 'loops', is seen as costly and to be avoided. It generally requires wasteful rework of earlier phases.

One reason for project errors and consequent loops is that the problem being solved is too large or too uncertain. If it is too large, then it is advisable to break it up into smaller stages and build the system a stage at a time.

Staged design can only be used where it is possible to break a problem up into a number of distinct subsystems. Thus, the first step may be a feasibility study that determines the size of a project. The feasibility steady may suggest that the project is very large and that it should be broken up into stages. If staged design is approved, the next step will be to break the whole project up into a number of stages. Each stage is then developed using the linear cycle and each subsystem is developed separately.

Structure Systems Development

Structured analysis is used to define and describe the system that best satisfies user requirements, given certain time and budget constraints. The objective of structured design is to minimize the lifetime cost of an information system by emphasizing maintainability, because maintenance is the costliest component in a system's life cycle. The critical point in design is to match the solution to the problem.

Structure Tools for Analysis

We require three types of new analysis phase tools:

- Something to help us partition our requirement and document that partitioning before specification. For this we use a Data Flow Diagram (DFD) a network of interrelated processes.

- Some means of keeping track of and evaluating interfaces without becoming unduly physical. For our interface tool we adopt a set of Data Dictionary convention, tailored to the analysis phase.
- New tools to describe logic and policy, something better than narrative text. For this there are three possibilities: Structured English, Decision Tables, and Decision Trees.

Structured Analysis

Structured Analysis is the use of these tools:

- Data Flow Diagram.
- Dictionary convention.
- Structured English.
- Decision Tables.
- Decision Trees.

Target Document:

Establishes the goals for the rest of the project. It says what the project will have to deliver in order to be considered a success. The target document is the principal product of analysis.

Models in Structured Analysis and Design

In the structured analysis process, emphasis is on the logical components of the system rather than its physical components. Consideration of design and implementation issues is postponed until agreement has been reached between designers and users on the function or objectives, of the system. This analysis is performed according to the following models:

Model 1:

Describe how the current system (old system) works by depicting its physical components.

Model 2:

Show the logical operations performed by the current system through logical abstraction of the current system operations.

Model 3:

Planning for the new system, by adding general physical components to the logical model of the new system we then arrive at a physical model of the new system

Model 4:

Prepare figures and diagrams that illustrate the analysis, design and actual implementation of the new system.

Gathering Information

Search Strategies

A search strategy is formulated by two important choices.

First, sources from which information is to be obtained are identified together with methods to obtain the information from each source.

Second, a search procedure for getting the information is established. The search procedure defines where to begin the search and how to continue. It also identifies the sequence in which sources will be searched and what information is to be gathered at each step.

The main elements of a search strategy

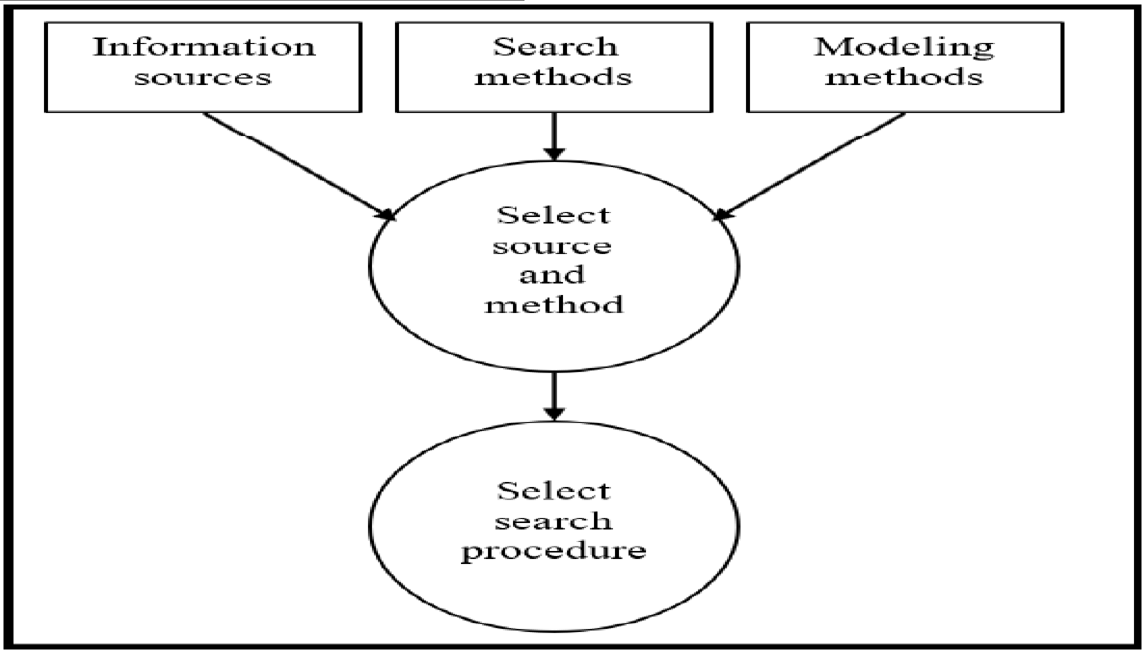


Figure 11: The main elements of a search strategy

Information Sources

Below are some common sources of information and appropriate research methods.

- **System user**

  They are usually the first source of information that analysts look for. From users it is possible to find out the existing system activities and to determine the user objectives and requirement. The usual search methods here are interviews or sometimes questionnaires.

- **Forms and documents**

  It is a useful source of information for system date flows and transactions. The search method begins with the analyst obtaining a list of such documents from the system users. Analysts then go through the documents to find their date elements and date structures.

- **Computer programs**

  They can be used to determine the details of date structures or processes. The search methods are often laborious and involve reading the program or its documentation and sometimes running the program with test date to see what it does.
- **Procedure manuals**

  It helps in determining what people do in the organization. Analysts to determine detailed user activities can use them. This detail becomes important in detailed system design. The method again calls for a detailed examination of the techniques of the applicable design methodology.
- **Reports**

  Reports indicate the types of outputs that users need. They can be used as a basis for user interviews.

Search Procedures

The search procedure has a number of general requirements. First, it must be top-down in nature. It must also ensure that all information sources are examined to ensure that all information about the system is gathered.

The search procedure should specify the organizational level at which interviews start, the personnel to be interviewed and any other information sources to be used. They should also include an interview plan and times when other sources are to be examined.

Search procedures very depending on whether an existing system is being investigated or whether a totally new system is to be designed.

Search Methods

There are two important search methods, interviewing techniques, and questionnaires.

➢ **Interviewing**

It is a process that is used by the analyst to gradually build a model of the system and to gain understanding of any systems problems. The interview plan, it specifies:

- The users to be interviewed.
- The sequence in which users are interviewed.
- The interview plan for each user.

➢ **Questionnaires**

The questionnaire contains a set of questions that the analyst has set for the user to answer, and the analyst analyzes those answers.

Questionnaires are useful for gathering numerical data or getting relatively simple opinions from a number of people, but they are not very effective for in- depth searches or identifying system problems or solutions. Interviews tend to be more successful for this purpose.

Starting a Project

The first two linear cycle phases, *problem definition* and *feasibility analysis*, are used to start a project. These phases are important because they set the direction for the remainder of the project. The first phase defines the project goal. The second determines a feasible way for achieving this goal. Feasibility analysis usually considers a number of project alternatives, one of which is chosen as the most satisfactory solution.

Setting the Project Goal

Project goals have two important aspects for systems analysts: how they look and how they are specified. There are many ways of setting project goals.

One important guideline for goal setting is to remember that goals should not be unrealizable ideals that are subsequently ignored. They must be developed within the practicalities of the organization. One way to ensure that such practicalities can be met is to elaborate the project goal into more detailed sub-goals that consider organizational constraints. These sub-goals are used in later stages to guide detailed analysis and design.

Another guideline for goal definition is the identification of deficiencies in the existing system. The project goal is to remove sub-deficiencies. Deficiencies are usually found through interviews or by examining documents about system performance. During initial analysis, therefore, interviewers should search for deficiencies such as:

- Missing functions;
- Unsatisfactory performance; or
- Excessively costly operation.

So far, we have suggested that goals are found by examining the internal operation of a system. Examining a system’s external environment can also set goals. One important external source is competitor performance. It is always worthwhile to examine what similar organizations do and see whether some of their methods can be used in the proposed system. Technical developments outside an organization are also an important source, especially for systems that use computers. New technical developments must be evaluated to see if they can improve system operation.

Generating Broad Alternative solutions

Feasibility analysis begins once the goals are defined and agreed upon. It starts by generating broad possible solutions, which are used to give an indication of what the new system should look like. At this stage there is no need to go into the detailed system operation.

The solution should provide enough information to make reasonable estimates about project cost and to give users an indication of how the new system will fit into the organization.

Two things must be kept in mind when proposing a broad solution. It should give people a good idea of what the new system will look like and also convince them that it will work. Another objective of the broad solution is to form an estimate of cost. To do this, a broad solution should include the major parts of a proposed system.

Components of a Broad Solution

The broad solution should include any additional equipment that will have to be purchased for the project, in order to estimate some of the direct costs of the project.

The broad solution should also specify what is to be done by the computer and what will remain manual. From this we can find the amount of data to be stored on the machine and any transmission costs of getting data to and from machine.

The solution should specify the information that will be made available by the system, only broad descriptions need be given here.

To investigate all possibilities, a number of broad solutions are proposed and evaluated to find the best solution. One guideline here is to start with three alternatives, namely:

- A totally automated system.
- A system with minimal automation.
- A system somewhere in between.

Any proposed solution should not exceed funding limits or ignore critical system operations or data needs.

Evaluating the Proposal

Once proposals are generated, they are evaluated. Three things are done in the evaluation:

- The technical feasibility of a proposal is evaluated. Is the technology needed for the proposed system available? If it is available, can it be used in the organization?
- The operational feasibility is looked at. Can the proposed system fit in with the current operations? Does it provide the right information for the organization's personnel? Does this information arrive at the right place in time?
- The economic feasibility is to be considered. Is it worthwhile to pursue the project? Or will the amount of money needed to implement it never be recovered?

Economic Feasibility

To carry out an economic feasibility study, it is necessary to place actual money values against any purchases or activities needed to implement the project. It is also necessary to place money values against any benefits that will come from a new system created by the project. Such calculations are often described as cost-benefit analysis.

Cost-benefit Analysis

Cost-benefit analysis usually includes two steps. One is to produce the estimates of costs and benefits. The other is to determine whether the project is worthwhile once these costs are ascertained.

Producing Costs and Benefits

The goal is to produce a list of what is required to implement the system and a list of the new system's benefits.

Cost-benefit analysis is always clouded by both tangible and intangible items.

Tangible items are those to which direct values can be attached (e.g. the purchase of equipment, time spent by people writing programs or items such as insurance costs or the cost of borrowing money). Some tangible costs often associated with computer system development are:

1. Equipment costs for the new system. Various items of computing equipment as well as items such as accommodation costs and furniture are included here.
2. Personnel costs. These include personnel needed to develop the new system and those who will subsequently run the system when it is established. Analysts, designers and programmers will be needed to build the system. Also included are any costs incurred to train system users?
3. Material costs. These include stationary, manual production and other documentation costs.
4. Conversion costs. The costs of designing new forms and procedures and of the possible parallel running of the existing and new systems are included here.
5. Other costs. Sometimes consultant's costs are included here together with management overheads, secretarial support, travel budgets and so on.

Cost of the system

The sum value of costs of items needed to implement the system is known as the cost of the system.

Benefit of the new system

The sum value of the savings made is known as the benefit of the new system. Once we agree on the costs and benefits, we can evaluate whether the project is economically viable.

The cost estimates are usually used to set the project budget. Often it is convenient to divide these costs into project phases to give management an idea of when funds and personnel will be needed. The cost estimates need to be worked out very carefully. One should avoid any omission from the estimates necessitating requests for more funds because something was forgotten.

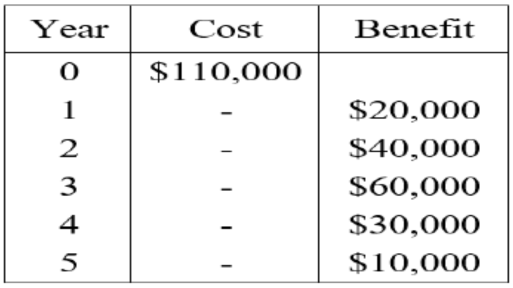
Determining Whether a Project is Worthwhile

The costs and benefits are used to determine whether a project is economically feasible. There are two ways to do this, the payback method or the present value method.

Payback Method

The payback method defines the time required to recover the money spent on the project.

As an example, consider the Table, which show the cost of implementing a project in the year 0 to be $111,000. Benefits from the project are obtained over the next five years, with the actual benefits also shown in the Table. If you add the values in the benefit column, you will note that the original $111,000 invested is returned some time towards the end of year 3. Hence the payback period is about 2.9 years.



Year	Cost	Benefit
0	$110,000	
1	-	$20,000
2	-	$40,000
3	-	$60,000
4	-	$30,000
5	-	$10,000

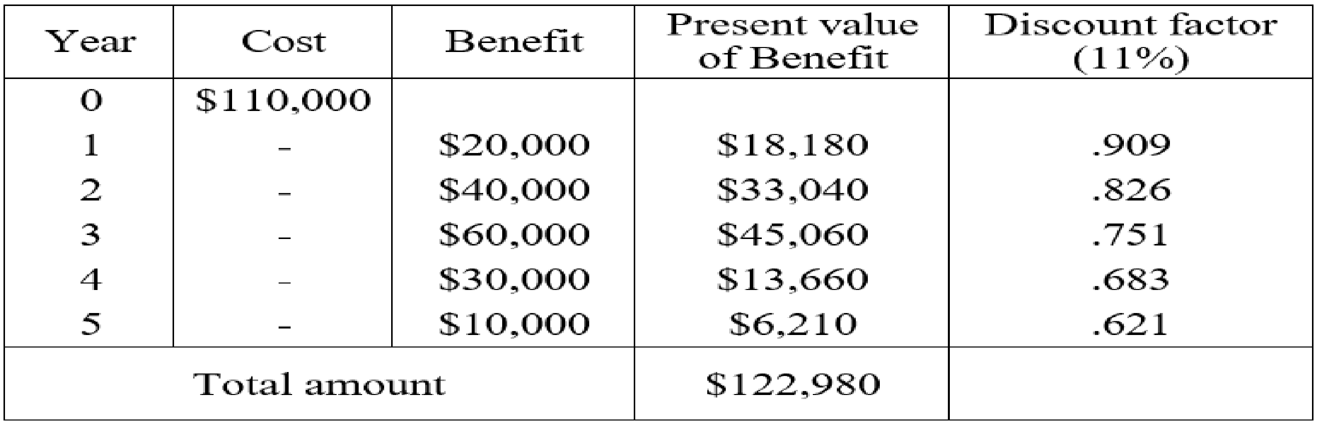
The present value method

The payback method is not always the best way to determine economic feasibility. To some extent the present value method works backwards. First, the project benefits are estimated for each year from today. Then, we compute the present value of these savings. If the project cost exceeds the present value the project is not worthwhile.

Let as look at the Table to see whether the project is worthwhile at a discount rate 10 percent. To do this we find the present value of the benefit at each year. The formula is:

Present value at Year_n = Benefit at Year_n / $(1 + {}^{r}/_{100})^n$

We call $1/(1 + {}^{r}/_{100})^n$ the discount factor in Table



Year	Cost	Benefit	Present value of Benefit	Discount factor (11%)
0	$110,000			
1	-	$20,000	$18,180	.909
2	-	$40,000	$33,040	.826
3	-	$60,000	$45,060	.751
4	-	$30,000	$13,660	.683
5	-	$10,000	$6,210	.621
Total amount			$122,980	

Thus, for example, the present value of $40,000 benefit at Year 2 is computed as:
Present value at Year2 = Benefit at Year2 / (1 + 10/100)2 = 33,040
Note that the sum of the present value in the Table is $122,980. As this exceeds $110,000, the project is worthwhile.
You may wish to compute the present value of the benefits in the Table at 15% interest.

Data Flow Diagrams

Data Flow Diagram (DFD)

A data flow diagram (DFD) is a network representation of a system. The system may be automated, manual or hybrid. A data flow diagram depicts the system in terms of its components which are:

- The system processes.
- The data used by these processes.
- Any external entities that interact with the system.
- The information flows in the system.

Data Flow Diagram Elements

Data Flow Diagrams are made up of only four basic elements:

1. **Data Flow Lines:** represented by named vectors.
2. **Processes:** represented by circles or "bubbles".
3. **Files:** represented by straight lines.
4. **Data sources and sinks:** represented by boxes.

Example:

We can read this data flow diagram as follows: -
"X's arrive from the source S and are transformed into Y's by the process P1 (which requires access to the file F to do its work). Y's are subsequently transformed into Z's by the process P2."

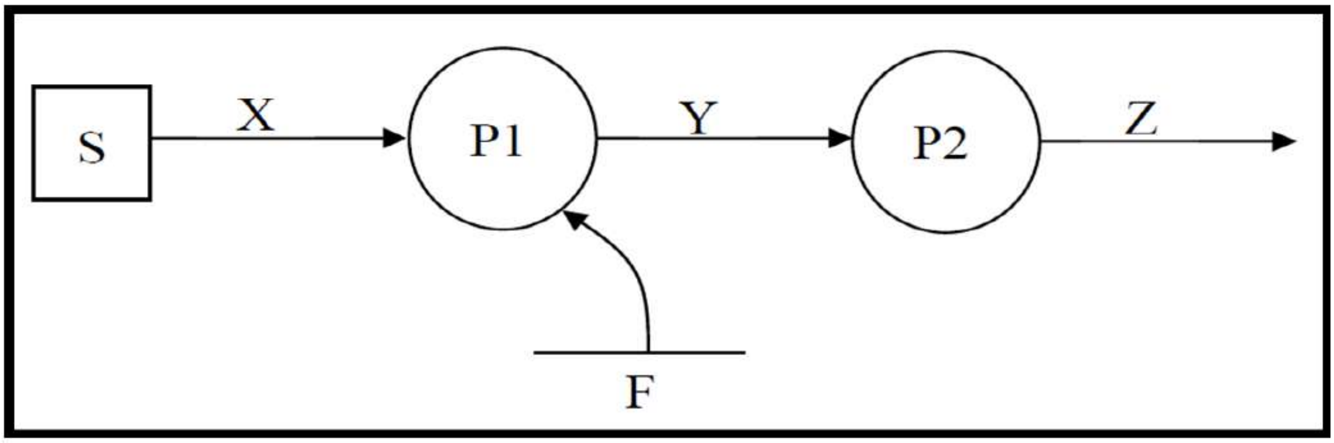


Figure 12: Example of data flow diagram

Structure Charts

a structure chart reveals both the modular structure of a system (i.e., it partitions into modules) and the hierarchy into which the modules are arranged. In addition, a structure charts also shows the data and control interfaces among modules. It does not, however, show the decision structure of a system except the major decision(s).

Structure Chart Elements

The elements of a structure chart-namely:

a) Module.
b) Connections.
c) Couples.
d) Loops.
e) Decisions.

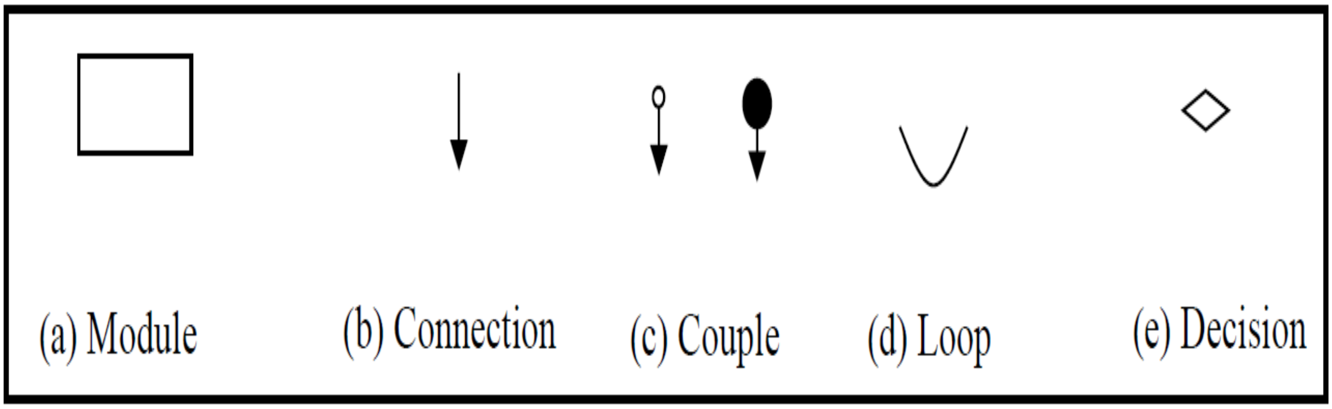
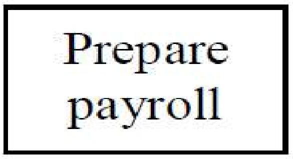


Figure 13: Structure Chart Elements

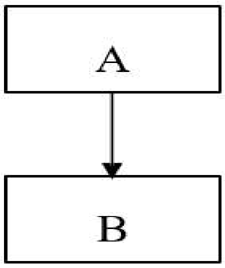
Example:

The elements of structure chart and their respective symbols are summarized as:

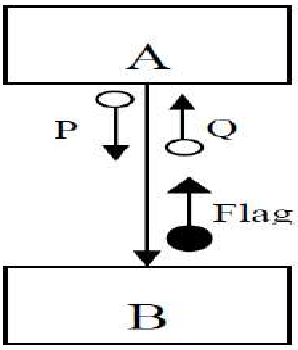


A module named "Prepare payroll"

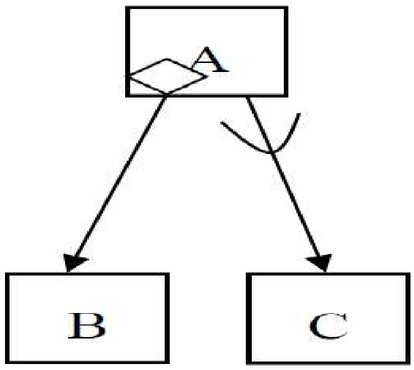
اسم الموديل هو اعداد المرتبات



الموديل **A** يستدعي الموديل **B** ورغم أن السهم أحادي الاتجاه، فمن المفهوم أن الوحدة المستدعاة تعيد التحكم إلى الوحدة التي استدعتها



الموديل **A** يستدعي الموديل **B** ويمرر له عنصر البيانات **P** ويستقبل منه عنصر البيانات **Q** وعنصر التحكم **Flag**



الموديل **A** يستدعي الموديل **B** من داخل جملة شرط **(if)**
ويستدعي الموديل **C** من داخل دورة تكرارية **(loop)**

Example

A payroll preparation function where employee pay record retrieval, gross pay computation, net pay computation, and pay check printing is considered to be the main modules.

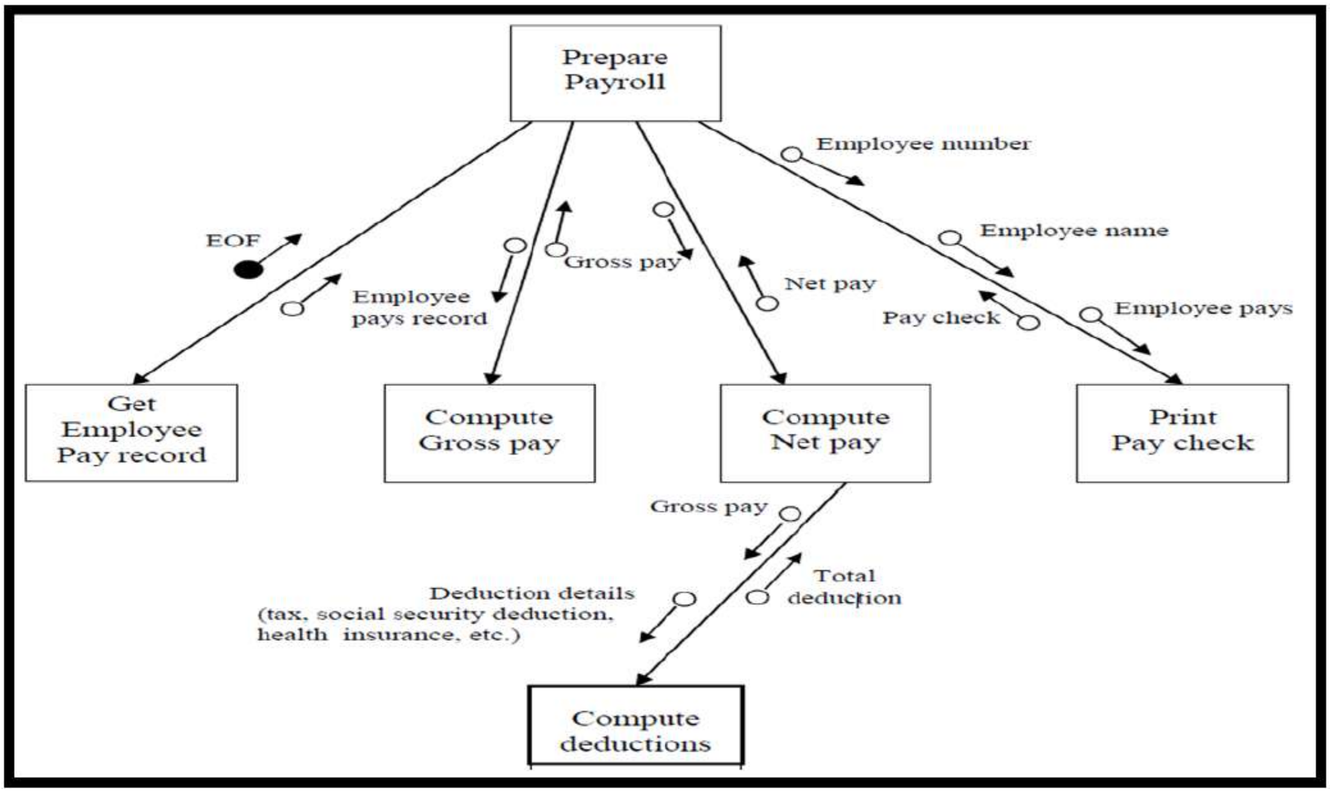


Figure 14: payroll preparation function

Data modeling

Data modeling is made up of more than one level. At the first level, analysts develop a conceptual model of data. This model represents the major data objects and the relationships between them. The next level organizes this model into a good shape. Usually this means removing redundancies, a process often called normalization, which uses ideas from relational theory. This normalized model is then converted to a physical database.

Conceptual Modeling

A conceptual model describes the essential features of system data. Just like a DFD, a conceptual model consists of a number of symbols joined up according to certain conventions. We will describe conceptual modeling using symbols from a modeling method known as entity-relationship analysis.

Entity-Relationship Analysis (E-R)

It uses three major abstractions to describe data, these are:

- Entities, which are distinct things (or data objects) in the enterprise.
- Relationships, which are meaningful interactions between the objects.
- Attributes, which are the properties of the entities and relationships.

The idea of E-R modeling is to group similar objects or thing into entity sets. The most common system entities are distinct physical things in the organization, thing such as persons, parts, projects and so on. Thus, all persons would fall into one entity set named PERSONS, all parts into one entity set named PARTS and so on. In any case, you would not place persons and parts into the one entity set.

After defining entity sets, we look at how entities in the entity sets can interact with each other and model this interaction by relationship sets. Thus, once we find out that persons work on projects, we would add a relationship set between the PERSONS and PROJECTS entity sets and give this relationship an appropriate name.

Modeling Relationship Cardinality

One property often shown on E-R diagrams is relationship cardinality, which specifies the number of relationships in which an entity can appear. An entity can appear in:

- In one relationship.
- Any number of relationships (N).
- A maximum number of relationships.

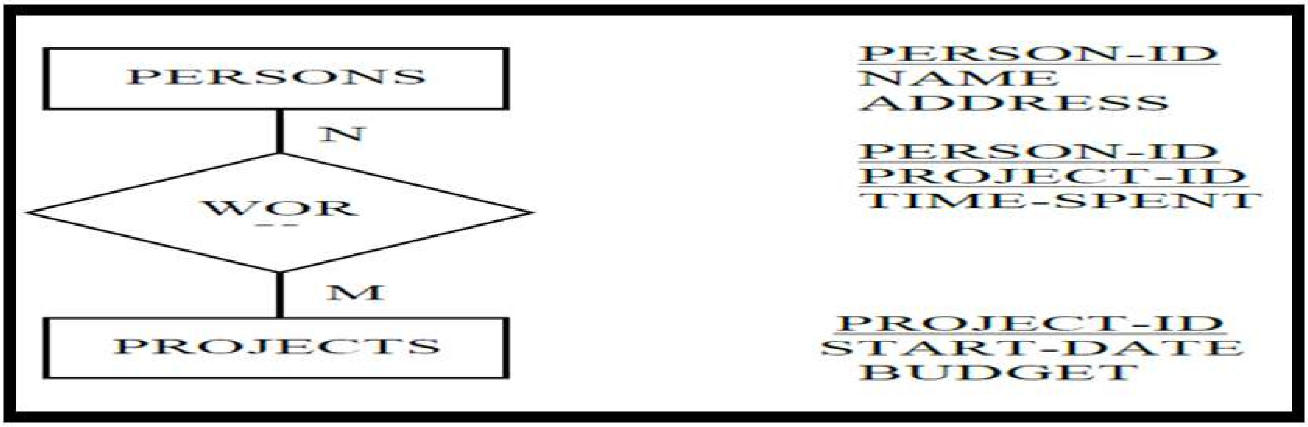


Figure 15: Adding cardinality

As you see in the Figure 15 a person can appear in more than one WORK relationship and so can a project. The letters N and M shows this on the E-R diagram.

These letters stand for variables and can take any value. If there was a limit to the number of times an entity can take part in the relationship, then N or M would be replaced by the actual maximum number. N and M are used to show that entities in the different sets may participate in a different number of relationships.
The next Figure 16 illustrates a 1- N relationship. Here a project has one (1) manager, whereas a manager can manage any number (N) of projects.

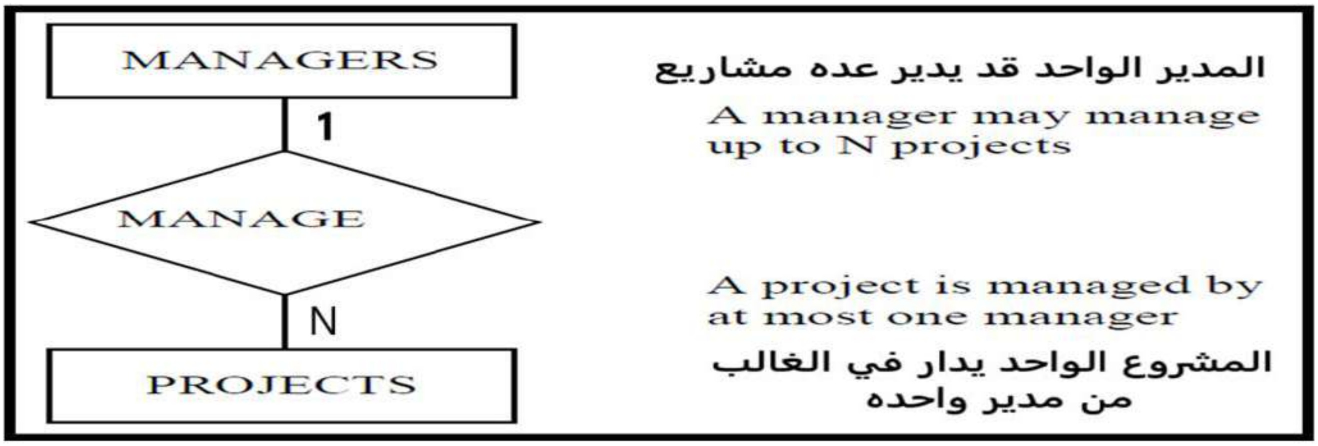

MANAGERS
1
MANAGE
N
PROJECTS
المدير الواحد قد يدير عده مشاريع
A manager may manage up to N projects
A project is managed by at most one manager
المشروع الواحد يدار في الغالب من مدير واحده

Figure 16: 1-N Relationship

<u>Modeling Relationship Participation</u>

E-R diagrams often specify the number of participation of entities in relationship. The participation of entities in a relationship set can be:

- Mandatory.
- Optional.
- Conditional.

If all entities in a given set must appear in at least one relationship in a relationship set, then, their participation in the relationship set is mandatory. If an entity need not appear in the relationship set, then its participation in the relationship set is optional. Conditional participation means that entities can only participate in a relationship if they have some specific attribute values.

In an E-R diagram, relationship participation is illustrating as a small circle on the link next to the entity set. There is no mark on the link for mandatory participation. A filled in circle on the link represents conditional participation.

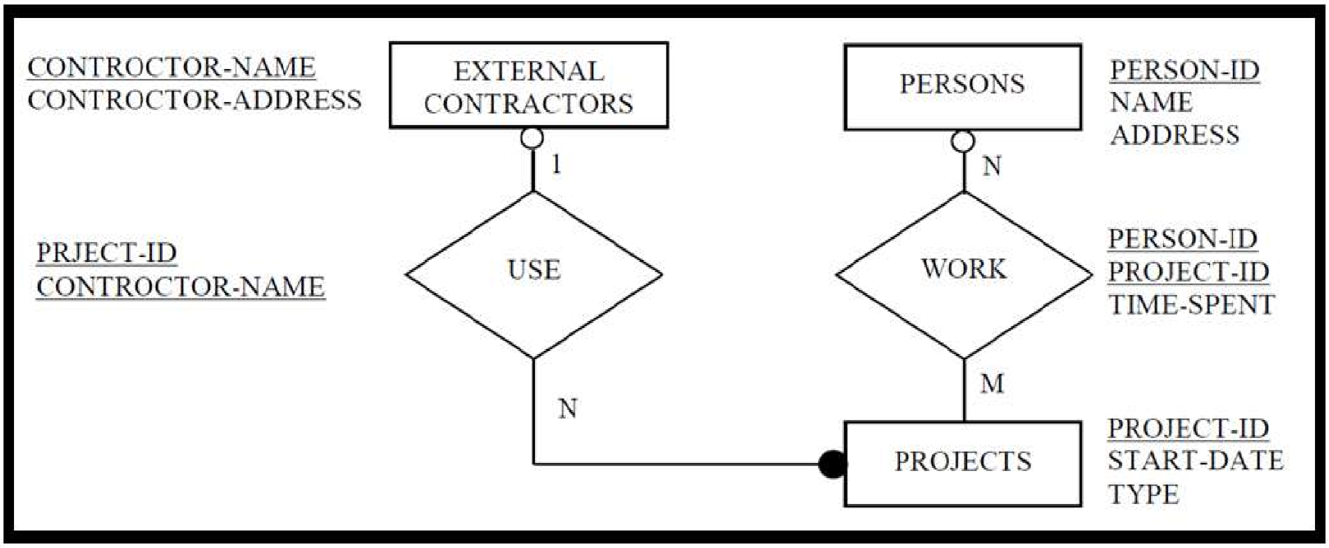


Figure 17: Relationship Participation

Binary and N-ary Relationships

So far, all the relationships in our E-R model have only had two participating entities. For example, relationship set WORK only had two participating entity sets, PERSONS and PROJECTS. It is also possible to have relationship sets with three or more participating entity sets. Relationships in such relationship sets are known as N- ary relationships. The Figure 18– is an E-R diagram of an N-ary relationship set, EXAMINES. This relationship has three participating entity sets, APPLICATIONS, EXAMINERS and TESTS. This relationship set models TESTS made by EXAMINERS on APPLICATIONS. There can be a number of tests made on each application. Each test identified by TEST-TYPE can be made by a number of examiners, who are identified by EXAMINER-NO.

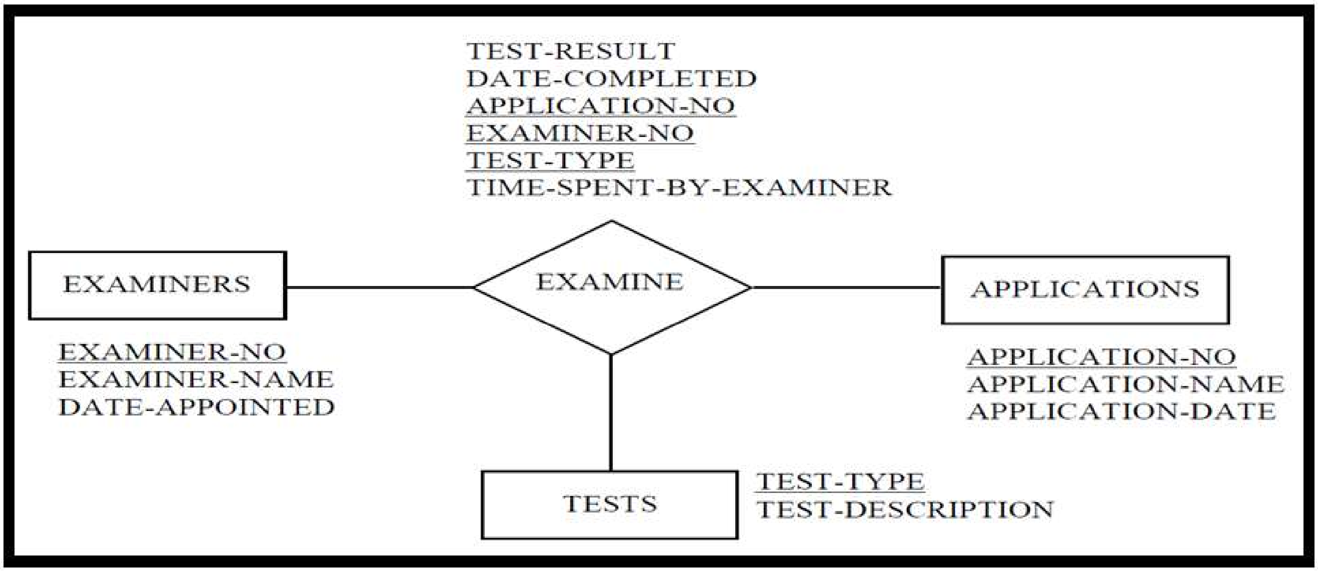


Figure 18: N-ary Relationship

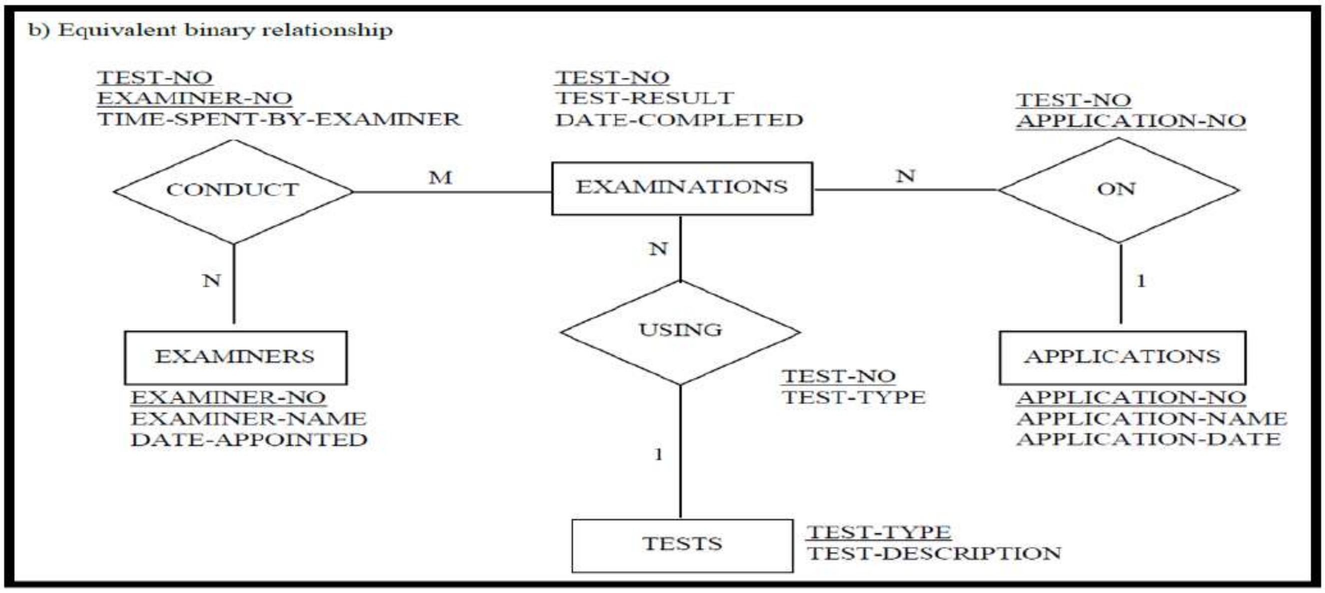


Figure 19: Converting an N-ary Relationship to Binary Relationship

There are a number of disadvantages in using N-ary relationships. First, you cannot specify cardinality on the links. If there are N examiners of one application and 1 examiner for each test, then, the link between EXAMINERS and EXAMINE would have to be labeled both 1 and N. secondly N-ary relationship sets are often not in BCNF.

Too many participating entities cloud the nature of the interaction. Perhaps when you have three interacting entities you can still see what is going on from the E-R diagram. However, when you get to five, six or more interacting entities, the representation may not be semantically clear, for this reason, there is something to be said for building models where all relationships have, at most, two interacting entities.

Converting an N-ary Relationship to a Binary Relationship

In practice, it is easier to begin by including N-ary relationships in the E-R model. Once the initial E-R model is constructed, all the N-ary relationship sets are replaced by relationship sets that are linked to at most two entity sets. There is a standard method for this conversion. In the standard method, the N-ary relationship is replaced by an entity set. Thus, the N-ary relationship set EXAMINE in Figure_19(a) becomes the entity set EXAMINATIONS in Figure_19(b). The entity set, which interacted in relationship set EXMINE, now interact in separate relationships (CONDUCT, USING and ON) with entity set EXAMINATIONS. It is common to give a special identifier to the newly created entity set. In Figure-19(b), the name of the new identifier is TEST-NO. some attributes are also moved during conversion. For example, some attributes of the N-ary relationship now become attributes of the newly created entity set or attributes of the new binary relationships. Thus, TEST-RESULT remains an attribute of the newly created entity set. Other attributes such as TIME-SPENT-BY EXAMINER, however become an attribute of relationship CONDUCT because they are dependent on the activity of the examiner of the test and not on the test as a whole.

Entity Modeling

One popular alternative to E-R modeling is entity modeling. Entity modeling does not distinguish between entity sets and relationship sets. Instead, everything is treated as entity sets.

The difference between an E-R model and an entity model is that, the E-R diagram (Figure 20: a) shows a number of entities set, also there are a number of relationships sets. In an entity diagram (Figure 20: b), everything is treated as entity sets. The links between the sets in the entity model are called associations and are the same as those in the E-R model.

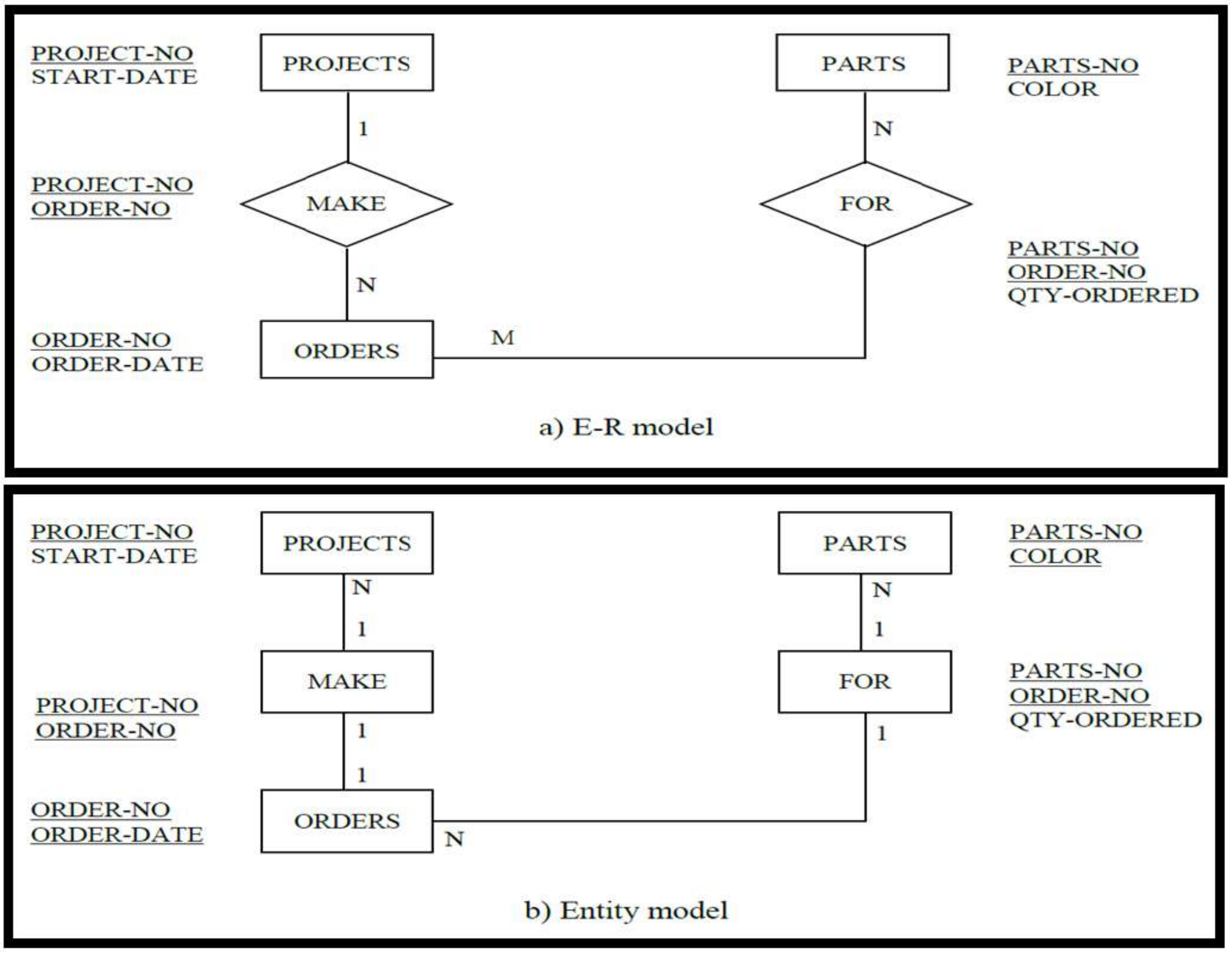


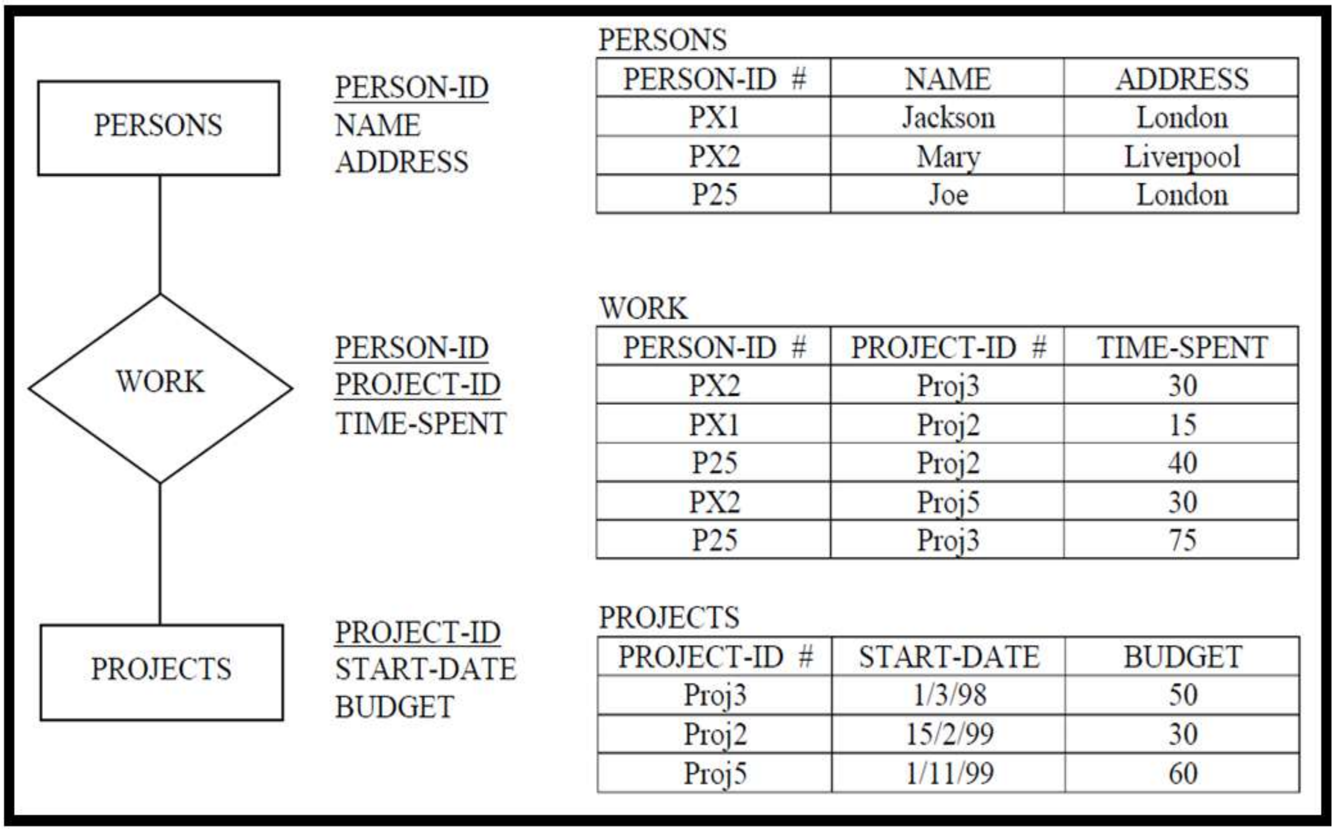
Figure 20: Comparing the E-R and Entity Models

You will also find that cardinality is labeled differently on the entity model. Cardinality shows how many times an entity in an entity set is associated with entities in another set.

You will find that data modeling using entity analysis proceeds in the same way as E-R modeling. We identify the entities and then add attributes to them, although it is no longer necessary to distinguish between entity sets and relationship sets. Any group of logically related attributes becomes an entity set.

The Next Level of Analysis

A conceptual model or an E-R model of a system is the first level of data analysis. We have to get the data model into shape by removing any redundancies from it. To do this, we use relational theory and replace each set in an E-R model by a table or relation (Figure 21). Thus, each entity set and each relationship set becomes a table or relation. The name of the set becomes the table name and the attributes of the set become the table columns.



PERSONS

PERSON-ID #	NAME	ADDRESS
PX1	Jackson	London
PX2	Mary	Liverpool
P25	Joe	London

WORK

PERSON-ID #	PROJECT-ID #	TIME-SPENT
PX2	Proj3	30
PX1	Proj2	15
P25	Proj2	40
PX2	Proj5	30
P25	Proj3	75

PROJECTS

PROJECT-ID #	START-DATE	BUDGET
Proj3	1/3/98	50
Proj2	15/2/99	30
Proj5	1/11/99	60

Figure 21: Replace each set in an E-R model by a table or relation

Note that entities in an entity set become rows in the relation that represents the entity set. Instead of tables we can use relational notation which only shows the relation's name and its attributes, as:

PERSONS (PERSON-ID, NAME, ADDRESS).
WORK (PERSON-ID, PROJECT-ID, TIME-SPENT).
PROJECTS (PROJECT-ID, START-DATE, BUDGET).

There are two reasons for converting the E-R model to a set of relations. First, such a conversion is a convenient step for going from an E-R model to a set of files. Each table eventually becomes a system file. The other reason is more important. There is a theory based on sound mathematics that specifies how we should construct tables to avoid data redundancies. What we are after is a set of 'normal' relations. If we organize the data into a set of such relations, we are guaranteed a good data design.

Normalization

A number of normal forms have been defined for relations. They are commonly known as:

- First normal form (1NF),
- Second normal form (2NF),
- Third normal form (3NF),
- Boyce-Codd normal form (BCNF); sometimes known as the optimal normal form (ONF),
- Forth normal form (4NF), and
- Fifth normal form (5NF).

Database designers must ensure that their relations are in the highest normal form.

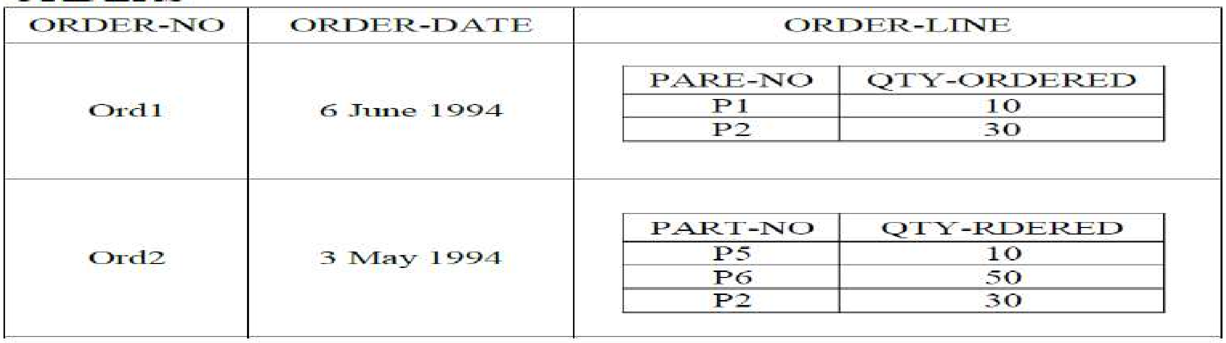
It should also point out that, although much of the work on relations has been of a mathematical nature, the beauty of the relational model is that the results of this mathematical work can be explained in terms relevant to practical database design. There are two reasons for this. First, there is the idea of flat files or relations. Normal relations must be flat, that is, all the column values must have simple values and cannot be groups of values. Flat file structures are both easier to access and to change in order to meet new user requirements. The second important criterion deals with redundancy – normal relations store each fact once only.

Normal Form and Non-normal Form Relations

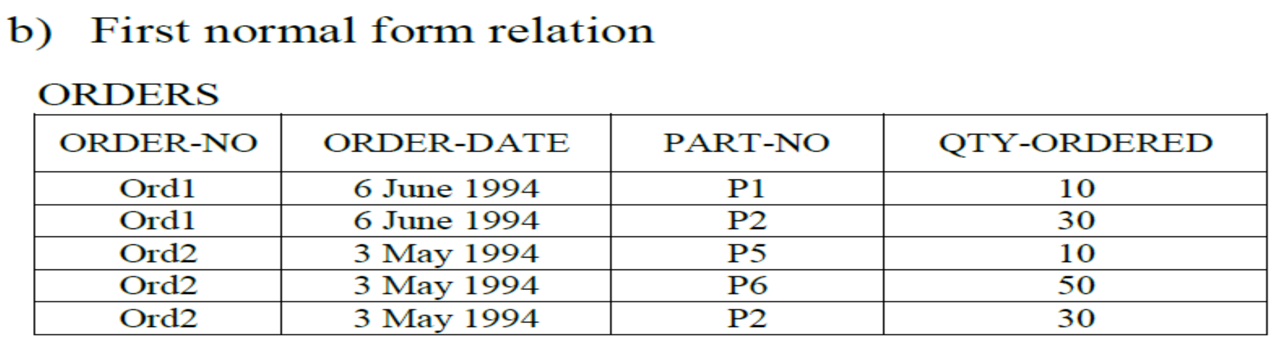
The difference between relations in normal form and those that are non-normal form is that, the attributes for the relation in non-normal form does not have simple values, but in normal form, attributes can only have one single value for one raw, thus, are said to have simple values. Relations that are not in normal form can always, be normalized. To do this, take each row in the unnormalized relation and look at the relation in the non-simple column, combine each row of the relation in the non-simple column with the values of other columns to make a row in the normalized relation. Repeat this combination for every row of the non-simple column until all of these values become normalized. Relations with only simple attribute values are in the first normal form (1NF) or are normalized.

a) Non-normal form relation

ORDERS



ORDER-NO	ORDER-DATE	ORDER-LINE	
		PARE-NO	QTY-ORDERED
Ord1	6 June 1994	P1	10
		P2	30
		PART-NO	QTY-RDERED
Ord2	3 May 1994	P5	10
		P6	50
		P2	30



b) First normal form relation

ORDERS

ORDER-NO	ORDER-DATE	PART-NO	QTY-ORDERED
Ord1	6 June 1994	P1	10
Ord1	6 June 1994	P2	30
Ord2	3 May 1994	P5	10
Ord2	3 May 1994	P6	50
Ord2	3 May 1994	P2	30

Figure 22: Non-normal form and first normal form relations

The important advantage of normalized relation is that each attribute has the same importance. This is useful when we convert the relation to computer software. It then becomes possible to add an index on any attribute field and retrieve data using any attribute as key.

Let us look at what characterizes higher relational forms. These forms must ensure that there is no redundancy in the data. To describe such higher normal forms, we have to define functional dependencies and relation keys.

Functional Dependencies

Functional dependencies describe some of the rules that hold between attributes in a system. In particular, they state whether a particular value of another attribute for that relation. That is a way of saying that if we know the value of X then we can determine a unique value of Y. A functional dependency is often expressed as:

$$X \longrightarrow Y$$

We can now say that attribute Y is functionally dependent on attribute X. Alternatively, we sometimes use the terminology that X determines a unique value of Y or that X determines Y.

Functional dependencies often involve more than one attribute on the left-hand side. Thus, to know the value of Z, we must know both the value of X, and the value of Y. The functional dependency then becomes: $X, Y \longrightarrow Z$

Derived Functional Dependencies

Sometimes, one functional dependency can be derived from other functional dependencies. For example, X may determine one value for attribute Z and suppose that each attribute of Z determines to one value of attribute Y. Given a value of X we once know the value of Z and from the vale of Z we can derive the value of Y. Mathematically we would say that if:

$$X \longrightarrow Z, \quad \text{and} \quad Z \longrightarrow Y$$

Then,

$$X \longrightarrow Y$$

It is up to the designer and analyst to ensure that there are no derived functional dependencies.

Relation Keys

A relation key is a set of columns whose values select unique relation rows. The idea of normal form relations centers around functional dependencies and relation keys. Informally, we would like the key attribute value of a relation to determine unique vale of the non-key attributes. One important thing to remember about relation keys is that they must apply for all possible contents of the relation.

Data Redundancies

Now let us consider a relation that stores some facts more than once. As explained before, the relation key of some relations is made up of two columns, the value of non-key attribute is determined by part of the relation key only and not by the combination of both parts of relation key. You should not that, in this case the non-key attribute is stored more than once. When facts are stored more than once, a relation is no longer in the highest normal form. Functional dependencies and relation keys can used to define second, third and optimal normal forms.

Second Normal Form

Relations where subset of key attributes determine non-key attribute values may be first normal form but cannot be in second normal form. To be in second normal form, a relation must not have a non-key attribute that is functionally dependent on part of the relation key.

Relations that are not in second normal form can be decomposed into second normal form relations. For example, the relation ORDERS can be decomposed into two relations, relation ORDERS and ORDERS-CONTENTS. You will note that in these relations non-key fields are determined by the whole (and not part of) relation key. Relations with this property are said to be in second normal form.

ORDERS

ORDER-NO #	ORDER-DATE
Ord1	6 June 1994
Ord2	3 May 1994

ORDERS-CONTENTES

ORDER-NO #	PART-NO #	QTY-ORDERD
Ord1	P1	10
Ord1	P6	30
Ord2	P5	10
Ord2	P6	50
Ord2	P2	30

Third Normal Form

Finally, it is also possible to have non-key fields are functionally dependent on other non-key fields. Consider relation VWHICLES that is stores information about cars. The values of the non-key attributes are determined by the whole relation key and the relation is in second normal form.

RELATION-KEY

VEHICLES

REGISTER-NO #	OWNER	MODEL	MANUFACTURE	NO-CYLINDER
Yx-01	George	Laser	Ford	4
Yj-77	Mary	Falcon	Ford	6
Yw-30	George	Corolla	Toyota	4
Yj-37	Mary	Laser	Ford	4
Yj-83	Andrew	Corolla	Toyota	4

However, you may note that there are functional dependencies between the non- key attributes. Relation VEHICLES to be in third normal form it should be decomposed into two relations as REGISTRATION and VEHICLE1, each relation:

- only contains facts about the relation key.
- have no facts between non-key columns.

RELATION-KEY

REGISTER-NO #	OWNER	MODEL	MANUFACTURE
Yx-01	George	Laser	Ford
Yj-77	Mary	Falcon	Ford
Yw-30	George	Corolla	Toyota
Yj-37	Mary	Laser	Ford
Yj-83	Andrew	Corolla	Toyota

Optimal Normal Form

All the relations discussed so far had one relation key. Normal forms also cover relations that have more than one relation key. The particular problem with relation that have more than one key is that part of one key may be functionally dependent on part of another key. For example, look at relation WORK, which has two overlapping keys. These are called overlapping keys because each key has PERSON-ID as a component.

RELATION-KEY

VEHICLE1

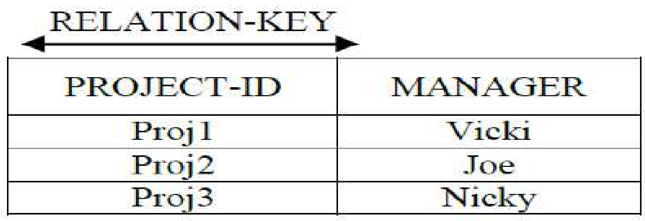
MODEL	MANUFACTURE	NO-CYLINDER
Laser	Ford	4
Falcon	Ford	6
Corolla	Toyota	4
Laser	Ford	4
Corolla	Toyota	4

However, using the strict third normal definition, relation WORK is in third normal form. What is needed is a higher normal form – one that takes overlapping keys into account. Such a higher normal form is the Boyce-Codd normal form

(BCNF). A relation R is in BCNF if for every functional dependency (X->Y) between any relation attributes, the attributes on the left-hand side are a relation key.

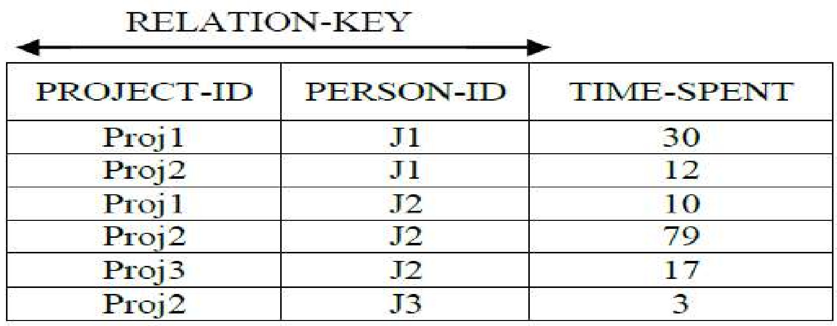
Relations where this property holds are in BCNF, which is sometimes called optimal normal form. The relation WORK is not optimal. Here there is a functional dependency: PROJECT-ID, MANAGER, but PROJECT-ID is not a relation key of WORK. Again, a non-BCNF relation can decompose into a set of optimal relations. The result of decomposing relation WORK is shown below:

PROJECTS


RELATION-KEY

PROJECT-ID	MANAGER
Proj1	Vicki
Proj2	Joe
Proj3	Nicky

WORK1


RELATION-KEY

PROJECT-ID	PERSON-ID	TIME-SPENT
Proj1	J1	30
Proj2	J1	12
Proj1	J2	10
Proj2	J2	79
Proj3	J2	17
Proj2	J3	3

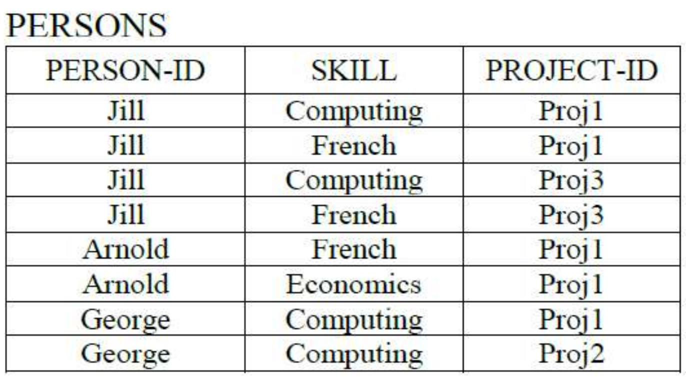
Normal Form Relations and Multi-Valued Dependencies

A multi-valued dependency exists in a relation when a value of one column or set of columns, X, determines a set of values in another column, Y, multi-valued dependencies are often expressed by the notation: $X \twoheadrightarrow Y$

Fourth Normal Form

Consider relation PERSONS, which is in 3NF or optimal form, SKILL and PROJECT-ID, are independent multi-valued dependencies of PERSON-ID. A person's skills are independent of the projects they work on, and the projects a person works on are independent of their skill.

Relation with dependent multi-valued facts


PERSONS

PERSON-ID	SKILL	PROJECT-ID
Jill	Computing	Proj1
Jill	French	Proj1
Jill	Computing	Proj3
Jill	French	Proj3
Arnold	French	Proj1
Arnold	Economics	Proj1
George	Computing	Proj1
George	Computing	Proj2

Therefore, a person's skills and person's projects should be stored in separate relations as shown in relation KNOWLEDGE and relation ASSGNMENTS bellow, which is in 4NF.

KNOWLEDGE

PERSON-ID	SKILL
Jill	Computing
Jill	French
Arnold	French
Arnold	Economics
George	Computing

ASSGNMENTS

PERSON-ID	PROJECT-ID
Jill	Proj1
Jill	Proj3
Arnold	Proj1
George	Proj1
George	Proj2

A relation is in 4NF if it does not contain independent multi-valued dependencies.

Fifth Normal Form

Fifth normal form (5NF) can be viewed as an extension of 4NF in the sense that now the multi-valued dependencies are no longer independent. For example, let as return to relation PERSONS. Suppose instead of containing information about a person's SKILL and PROJECT-ID, the relation also stores information about what skills that the person uses in a given project. Thus suppose 'Proj1' only needs 'Computing' and 'Economics' skill but not 'French'. The relation would now be as shown in relation APPLYING-SKILLS bellow:

Relation with dependent multi-valued facts

APPLYING-SKILLS

PERSON-ID	SKILL	PROJECT-ID
Jill	Computing	Proj1
Jill	Computing	Proj3
Jill	French	Proj3
Arnold	French	-
Arnold	Economics	Proj1
George	Computing	Proj1
George	Computing	Proj2

Relation APPLYING-SKILLS differs from relation PERSONS in that it does not contain rows that include both 'French' and 'Proj1'. This relation is in 4NF but still has undesirable properties in that:

- Some facts are stored twice.
- There are blank fields (a row with a NULL value).

These undesirable properties arise because APPLYING-SKILLS contains dependent multi-valued dependencies, that is, the value of SKILL, that is associated with PROJECT-ID depends on the SKILLS needed by the project. A property of a relation that is in 4NF but not in 5NF is that it cannot be decomposed into two relations but must be decomposed into three relations. Thus, to remove the undesirable properties in relation APPLYING-SKILLS, we must decompose it into the three relations shown below.

Process description

One of the major reasons for using structured systems analysis is to remove ambiguities from system descriptions. Such ambiguities frequently arise in natural language descriptions of processes.

The main techniques proposed for describing processes are:

- Structured English.
- Decision tables.
- Decision trees.

Structured English put verbal descriptions into a logical structure, which removes logical ambiguities. The next two methods (decision tables and decision trees) are preferred where one of a large number of actions is to be selected. The action selected depends on a large number of conditions. Structured English is not usually used for this purpose because the logic structure would become repetitive. The differences between the three methods are illustrated as:

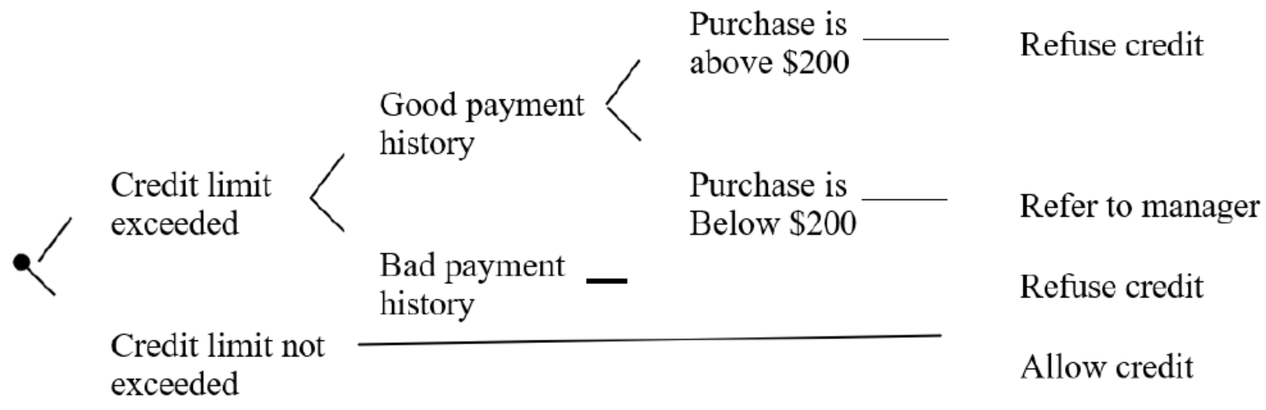
a) Using structured English

```
IF credit limit exceeded THEN
IF customer has bad payment history, THEN refuse credit
ELSE IF purchase is above $200 THEN refuse credit ELSE refer to manager
ELSE allow credit
```

b) Using a decision table

Credit limit exceeded	Y	Y	Y	Y	N	N	N	N
Customer has bad payment history	Y	Y	N	N	Y	Y	N	N
Purchase is above $200	Y	N	Y	N	Y	N	Y	N
Refuse credit	X	X	X					
Refer to manager				X				
Allow credit					X	X	X	X

c) Using a decision tree



Documentation

System documentation is an organized approach needed to keep track of system models (i.e., work produced during system analysis and design). This organized approach is needed for a number of reasons:

- First, it provides means for referencing model components and thus help to manage system complexity.
- Secondly, it supports maintenance as any work can be passed from one person to another.

Documentation is both a communication tool and a management tool. It is a communication tool because it contains a repository of all work done to date and makes it available to all persons working on a large project. Documentation in the first instance can be divided into project reports and a system description. In most organizations, the system description is kept in a system dictionary

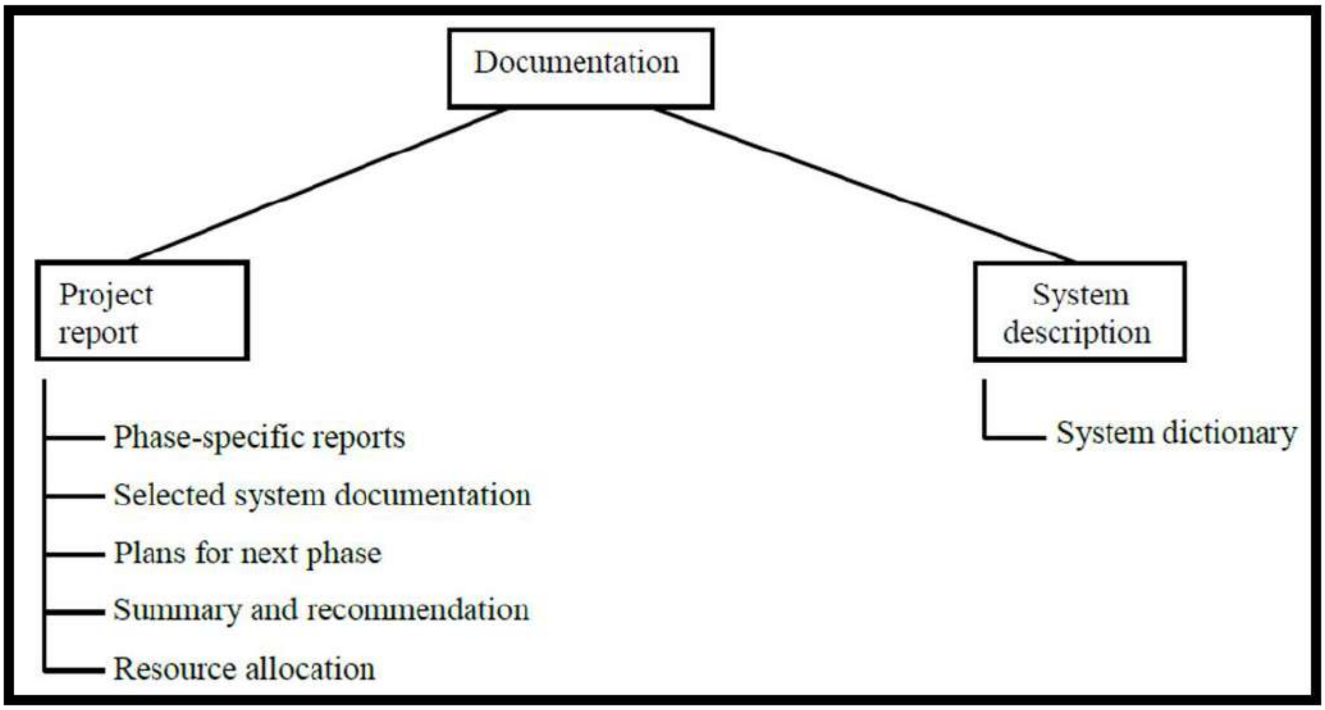


Figure 23: Documentation structure

Project Reports

Project reports include information required by project management. They are the deliverables expected at the end of each phase. Project reports include general as well as phase-specific information. General information includes a summary and recommendation for the next phase. It also includes a plan for the next phase, together with proposed resources for that phase.

Phase-specific information depends on the project phase. A project report at the end of a feasibility study will include different components to that provided at the end of systems analysis phase. The feasibility report will include expected project costs and a recommendation to proceed or abandon a project. The report at the end of systems analysis will describe the existing system and objectives for the new system. Project reports also include parts of the system description or references to parts of that description. In some cases, project reports may include high-level data flow diagrams (DFDs) and a high-level E-R model.

System Description

As shown in Figure 24, the structure of system dictionary is made up of a number of components. One important component is the DFD that describes the system. Another may be the E-R diagram that describes the system data. The process description component describes system processes and the data components, which is sometimes called the data dictionary. The user component describes system users and how they use the system.

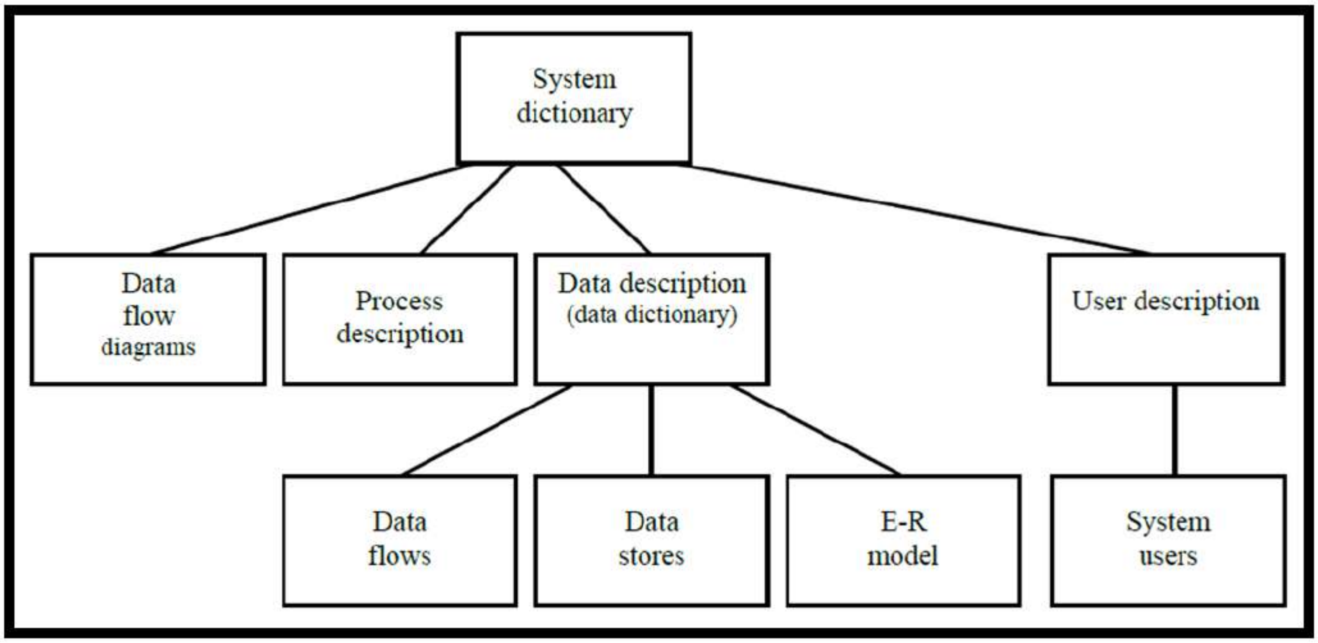


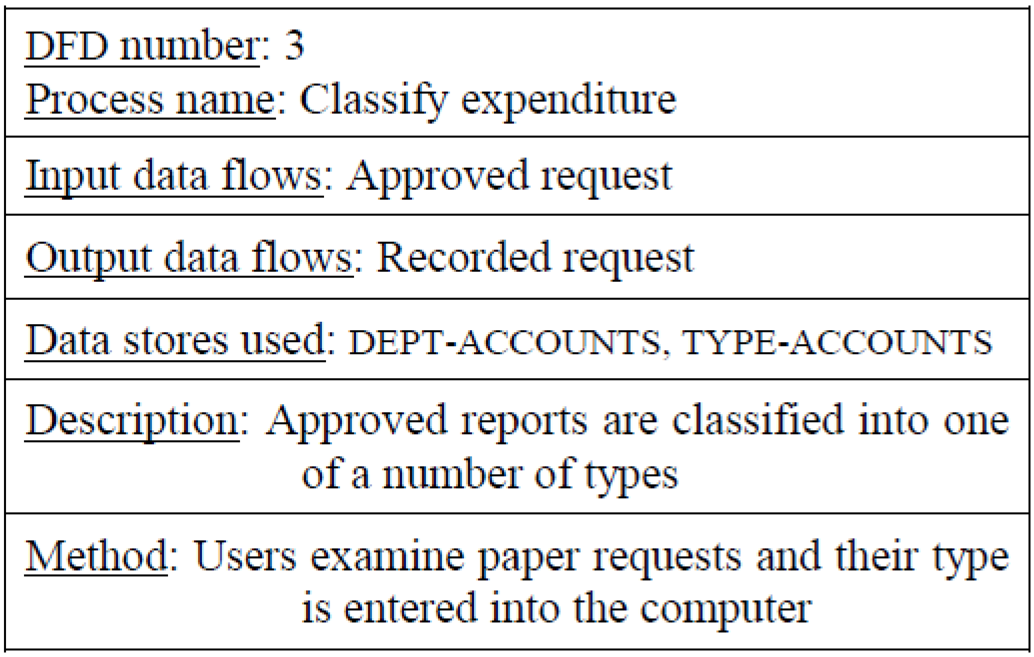
Figure 24: System description structure

Process Description

Process description includes an entry for each process in a data flow diagram. Each process entry includes the DFD number for the process together with the process description. The process description itself will depend on the process level.

High-level processes are usually described in an informal manner. They may be simple sentences that describe overall process goal. As shown in Figure 25, the entry includes the process number and name. It also includes the name of the data flows coming and out of the process and a narrative process description.

Low-level DFD descriptions are the same as those for the high-level process. Except that in this case, however, is more formal. Instead of narrative, the process description will now use structured English definitions or any of other process description methods.



DFD number: 3
Process name: Classify expenditure

Input data flows: Approved request

Output data flows: Recorded request

Data stores used: DEPT-ACCOUNTS, TYPE-ACCOUNTS

Description: Approved reports are classified into one of a number of types

Method: Users examine paper requests and their type is entered into the computer

Figure 25: Describing a high-level process

All the process entries are organized into their numeric sequence according to which they are entered into the system dictionary.

Data Dictionary

The components of a data dictionary in structured systems analysis are the data stores and data flows, also include the E-R model. Most data dictionaries also describe data elements and data structures separately from the data stores and flows. This save repeating the data element descriptions every time they occur in a data flow or store.

Instead of detailed data element descriptions, the data flow and store components and the E-R diagram description include cross-references to the components that they use. Figure 10.4 illustrates typical data dictionary entries and cross-references between them.

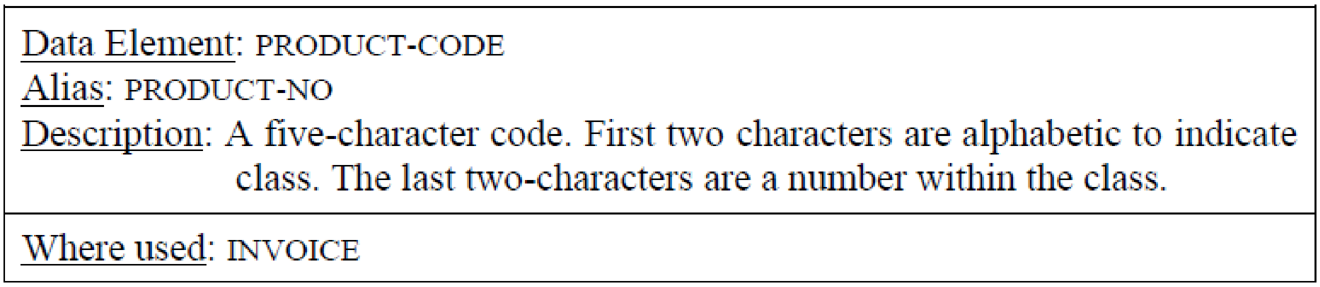
Describing data elements

There is one entry in the data dictionary for each data element. The entry describes any aliases or alternate names for the data elements. For example, in Figure 10.4, PRODUCT-NO can be used to mean the same thing as PRODUCT-CODE. The entry also includes the data element description, which includes the kind of values that the element can take and the range of these values.

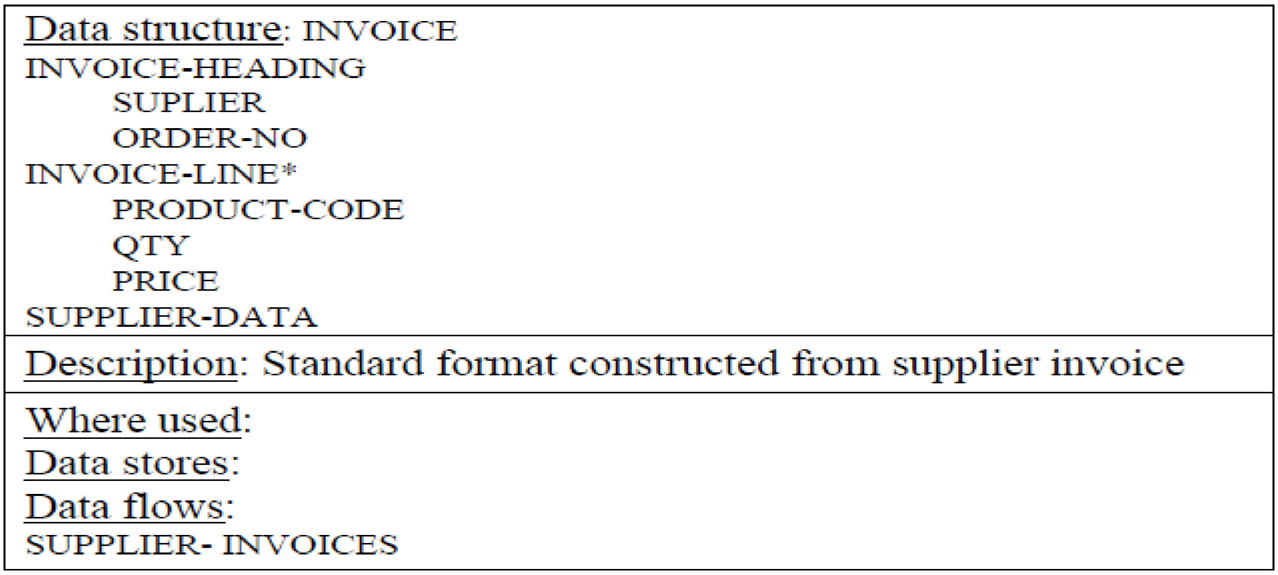
Describing structures

Data structures are combinations of data elements that appear in various parts of the DFD. Most data dictionaries describe structures by hierarchies. For example, the data structure described in Figure 10.4, is the INVOICE. The structure INVOICE is made up of a number of lower-level structures, namely, INVOICE-HEADING, INVOIVE-LINE and SUPPLIER-DATA. INVOICE-HEADING is made up of two data elements, SUPPLIER and ORDER-NO.

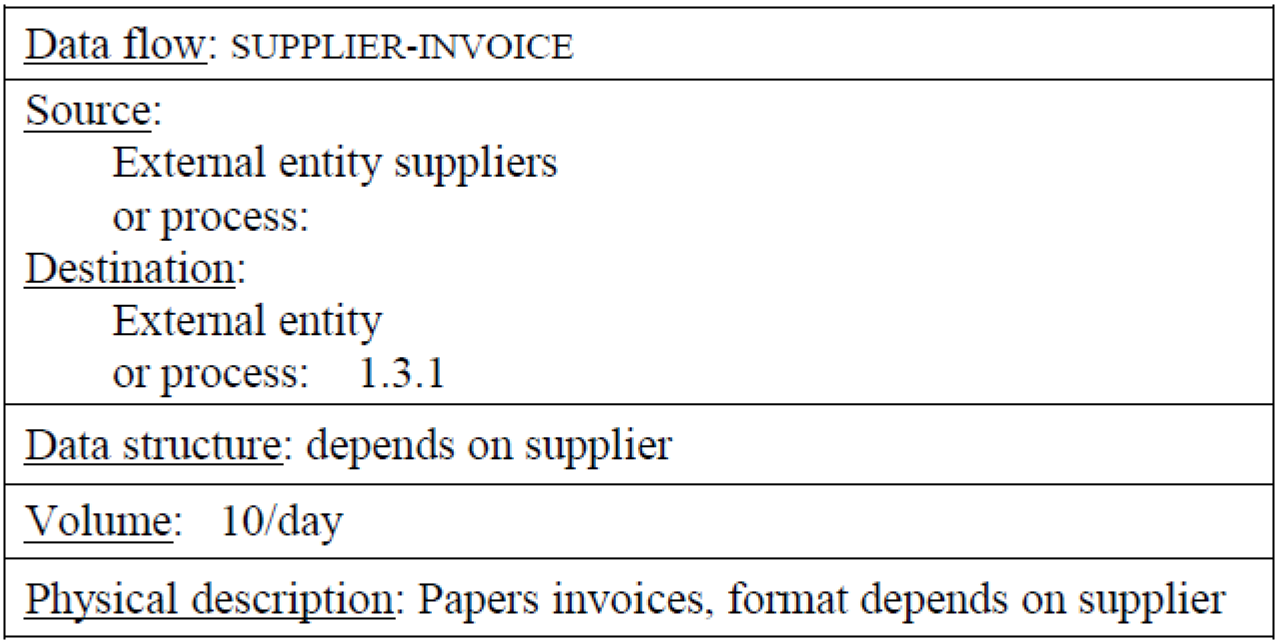
The illustration in Figure 26 uses a hierarchical description of structure. A hierarchy is described by listing the data elements that make up a structure as a hierarchy, for example:



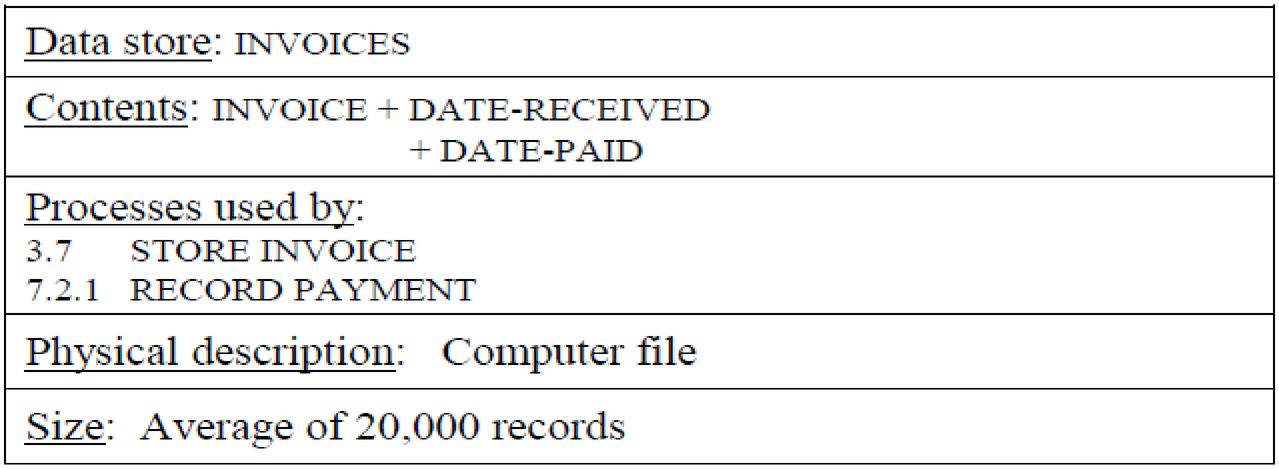
Data Element: PRODUCT-CODE Alias: PRODUCT-NO Description: A five-character code. First two characters are alphabetic to indicate class. The last two-characters are a number within the class.
Where used: INVOICE



Data structure: INVOICE INVOICE-HEADING SUPLIER ORDER-NO INVOICE-LINE* PRODUCT-CODE QTY PRICE SUPPLIER-DATA
Description: Standard format constructed from supplier invoice
Where used: Data stores: Data flows: SUPPLIER- INVOICES



Data flow: SUPPLIER-INVOICE
Source: External entity suppliers or process: Destination: External entity or process: 1.3.1
Data structure: depends on supplier
Volume: 10/day
Physical description: Papers invoices, format depends on supplier



Data store: INVOICES
Contents: INVOICE + DATE-RECEIVED + DATE-PAID
Processes used by: 3.7 STORE INVOICE 7.2.1 RECORD PAYMENT
Physical description: Computer file
Size: Average of 20,000 records

Figure 26: Data dictionary entries for structured system analysis

ORDER
SUPPLIER-NO
DATE-ORDERED
DATE-REQUIRED
ORDER-NO

Here ORDER is the structure name. The order is made up of four data elements, SUPPLIER-NO, DATE-ORDERED, DATE-REQUIRED and ORDER-NO. the indentation of the item names under ORDER implies that they are components of ORDER. Of course, most data structures are more complex than the ORDER structure shown above. Item values may be repeated, and there may be optional or alternative values and structures within structures. An example showing structures within structures and repetition follows:

```
ORDER
SUPPLIER-NO
DATE-
ORDERED
DATE-
REQUIRED
ORDER-NO
ITEM-
    ORDERED*
    ITEM-NO
    QTY-ORDERED
```

Now ORDER contains the structure ITEM-ORDERED. Because an order can contain many items the structure ITEM-ORDERED can be repeated many times. The asterisk (*) after the structure name indicates that a structure or a data element can be repeated. The notation also allows specification of alternative or optional items in a structure. This is done as follows:

```
  ORDER
{ SUPPLIER-NO }
  SUPPLIER-
  NAME DATE-
  ORDERED
  DATE-
  REQUIRED
  ORDER-NO
  [ORDER-
  STATUS] ITEM-
  ORDERED*
       ITEM-NO
       QTY-ORDERED
```

Here the curly brackets {} indicate alternative structures. Thus, an order may contain SUPLIER-NO or SUPLLIER-NAME but not both. The square brackets [ ] indicate an optional component. Thus, an ORDER may or may not contain the data element ORDER-STATUS.

There are of course, alternative ways of describing structures. One such alternative has been described by DeMarco. In this notation, structure components are described by using the plus sign (+) instead of indentations. Thus, ORDER is now described as:

Repetition is described by placing the repeating structure within curly brackets {}; alternative structures are defined by square brackets [] and optional structures by round brackets ( ).

Describing data stores and data flows

Data stores and data flows are made up of data structures and elements. Consequently, data store and data flow descriptions contain data structure and data elements. The data stores and elements of a DFD diagram can be described using any of the above notations. They also contain many other things. For example, as shown in Figure 10.4, the data flow description contains the origin and destination of the data flow. It also contains the physical description of the data flow, in this case stating that it takes the form of paper invoices.

Data store descriptions also contain physical information such as the medium used to store the data and where it is located.

Cross referencing

You should note the extensive cross-referencing between all entries in the system dictionary. There is cross-reference between data element entries and data structure entries with data element entries referring to the structures that contain the data element. Similarly, any data element in a data structure must have a data element entry.

The cross-reference from the structure to the data element is by the name of the data element.

Processes should reference the data stores and flows they use. Note that references to data stores are necessary because it is possible to have unlabeled flows between processes and stores. If we cross reference flows only, these unlabeled flows would not appear in the process entry. Cross-references from processes to data stores and flows let us know what data elements and structures the process uses.

Data stores include references to the data structures and elements that they use. Data stores also include references to the processes that the data store. Note that the reference is to the lowest order process. A cross-reference to process 1.3.1 also means that processes 1.3 and 1 use the data store.

Data flows reference their source and destination. They also reference the data elements and data structures that appear in the data flow. Also note that cross-referencing is through the use of a unique set of names. Thus, there is only one entry for PRODUCT-CODE in the data dictionary. Any structure, store or flow that includes PRODUCT-CODE also contains the name PRODUCT-CODE in its description. This name serves as the cross-reference.

Cross-referencing improves system maintenance. Whenever a change is proposed to one part of the system the repercussions of that change can be quickly evaluated. Furthermore, all parts of the system related to the change will be amended during the change, ensuring that no unforeseen errors arise when the change is implemented.

The Entity-Relationship Diagram and The Data Dictionary

The E-R diagram is developed in parallel with DFD diagram. There are entries in the data dictionary for each entity set and each relationship set.

There will be cross-references from these entries to the other entries in the data dictionary. Any time an attribute is identified as part of the E-R diagram, it is added to the data dictionary. Similarly, any data stores or flows that include a set in the E-R diagram will include cross references to that set. The set in turn will include a cross-reference back to the data stores or flows that contain the set.

System Users

System users can be individuals or organizational entries. The system dictionary will include details of these users and their responsibilities. It will also describe the data that the user’s access and the processes they use.

Maintaining A System Dictionary

The system dictionary can be manual or automated. It must include an entry for each data element, data structure, process description, user, data flow and store. The simplest way to maintain a system dictionary is to have a specific form for each type of dictionary entry. Each entry described in Figure 10.3 and 10.4 is entered on to paper sheet. This sheet is stored in a system manual, which is usually a loose-leaf folder. New entries can be added to the manual, but change to an entry requires the page for that entry to be replaced.

Automated Documentation

Manual system dictionaries are sometimes awkward to maintain. Every change requires a form to be replaced and if there is more than one copy of the system dictionary, this form must be replaced in each copy. Sometimes copies may not be updated and inconsistencies can arise. Many organizations are now beginning to use computer aids to maintain a system dictionary. A variety of such aids is possible, with each kind of aid giving a different level of functionality.

Computer Aids

Computer aids can go further in assisting analysts. They can guide analysts by guiding the modeling process and detecting any modeling inconsistencies. Thus, the inconsistencies are automatically brought to the analyst's attention, whereas in the manual case it is up to the analyst to detect them.
A computer aids can work in two ways. One is to rely on the analyst to input all the data about the model and then check the model for consistency. Another is for the computer to actively guide analysts in their work. This kind of aid can suggest work to be done next, identify missing pieces of the model and guide the designer by asking questions that fill in any model details.

Designing the new system

The first three phases of liner cycle describe methods, which is used to define projects, check project feasibility, and analyze systems. The next step of the cycle is to design the new system.

Problem Solving and Design

The problem-solving idea requires design to proceed in top-down way. Design usually proceeds in two steps, Phase 4A, a broad design is developed. Then, in Phase 4B, the broad design is expanded in to detail.

Broad design shows the parts of a system to be changed, and indicates the parts to be automated and those that will stay manual. Following agreement on the broad design, a detailed design is developed. A number of things are done during detailed design. The inputs are specified showing the layout of any input *screens* or *forms*. The outputs are also designed showing the layout of *reports* or *screens*. Thus, the presentation of deliveries by areas would be designed in this phase, as would inputs of trial schedules. In addition, the files, structures and programs are also specified, showing the different record types and program modules needed to make the system work. The detailed calculation needed to evaluate trial schedules would be specified at this point.

System Objectives

The objectives are the driving design factor. Any changes made to the system should be to satisfy these objectives. It is therefore important to state the objectives in a way useful to design.

Kinds of Objectives

There are many kinds of objectives, common types are:

- functional objectives that state new or amended functional system requirements. Included here can be:
  - new system functions requirements.
  - control over system maintenance.
  - security and access controls.
  - input and output methods.
- operational objectives that specify performance standards to be attained by the new system. These define system accuracy and various timing requirements.
- personal and job satisfaction needs. Important objectives here are designing system that an easy to use and which give users the ability to use their creative abilities rather than simply responding to computer outputs.

Different design ideas are needed to satisfy each of these objectives. The first objective can be achieved by adding new processes to the system or amending existing processes. It may also be necessary for new data to be stored in the system and for new data flows to be established. The operational objectives, on the other hand, require different solutions. Often these call for:

- a different physical realization of some of the system processes;
- a reorganization of the application of process transformations on a given data report.

Personal and job satisfaction objectives may call for changes to the user interface to the computer, new report layouts or changes to the data flows.

Reducing Objectives

Objectives can be high-level, low-level, general or specific. Usually, we start with high-level objectives and then reduce them as we proceed to detailed system analysis. Often objectives are defined by starting with *key performance factors*. Such key performance factors are general statements about how the goal will be met by the new system.

These key factors are then analyzed. To do this, analysts talk to the users to define precisely what each key goal entails. Usually, it is necessary to ascertain things like:

- the information requirements to satisfy the key goals.
- performance targets (such as time, volume, sample size).

- comparison to existing systems.
- human factors.

Once we have the objectives, design can commence. The design process described here uses the problem-solving steps proposed for structured system analysis.

Problem-Solving With Structured System Techniques

Proponents of structured system analysis have suggested a problem-solving approach for system design. This approach is illustrated in Figure 27. it is made up of four steps designed to determine:

- how the system works now.
- what the system does now.
- what the new system will do.
- how the new system will work.

Of course, the first two steps are unnecessary if you a building a completely new system. All the steps, however, apply if you wish to change an existing system.

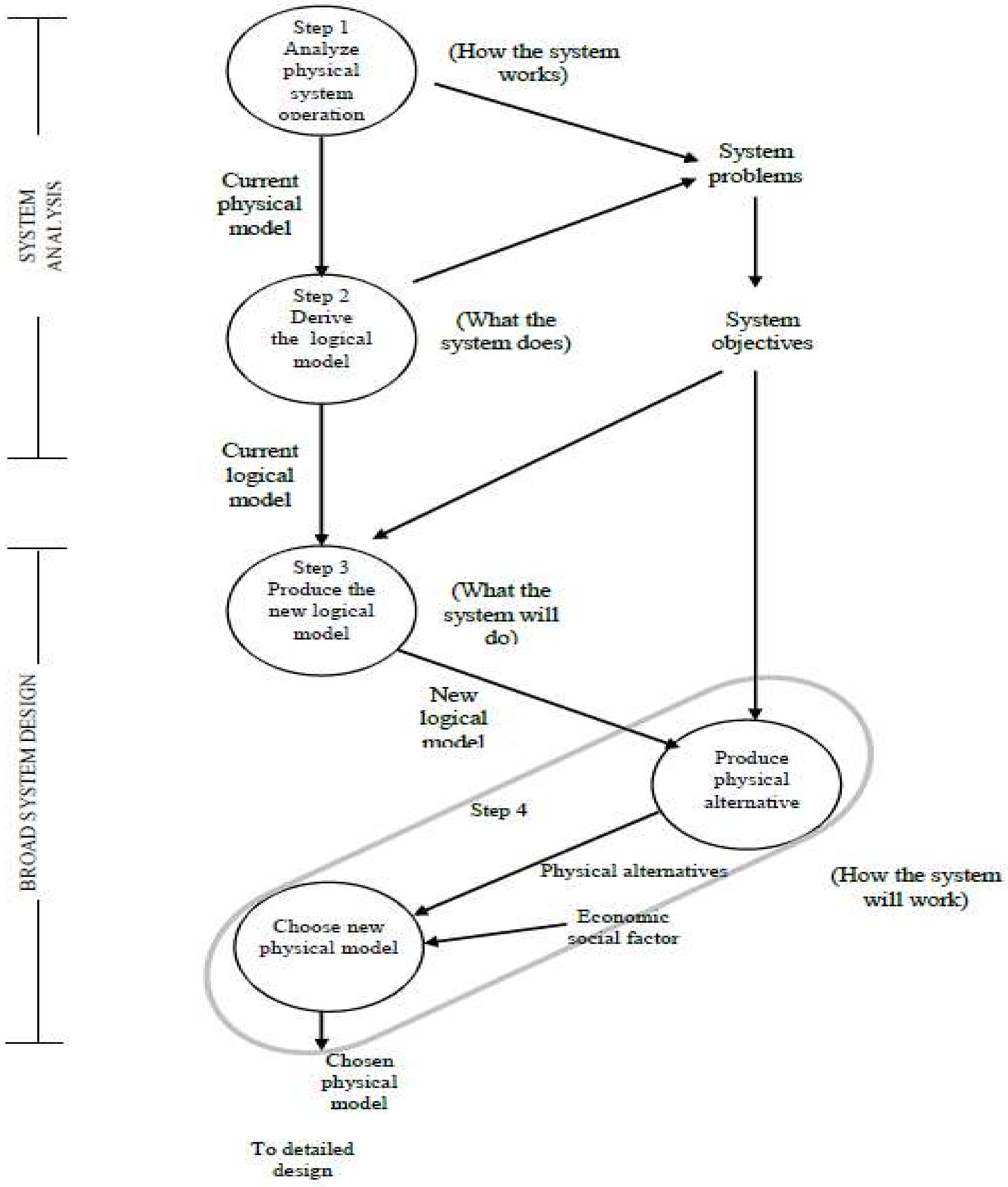


Figure 27: Problem-solving steps

The first two steps – System analysis

The first two steps in Figure 28 is to analyze the physical system operation and develop the current physical model, and covert the physical model to a logical model of the current system, that is, what the system does. System problems are also identified during these first two steps.

The next two steps – System design

Design commences once the logical model of the existing system is available. Design begins by using identified system problems to develop objectives for the new system. It then proposes a system that satisfies these objectives.

During step 3, the new logical model is developed. The new logical model includes any new processes or changes to existing processes necessary to meet system objectives. The designer must examine as many solutions as possible to make sure that a good solution is found to meet system objectives.

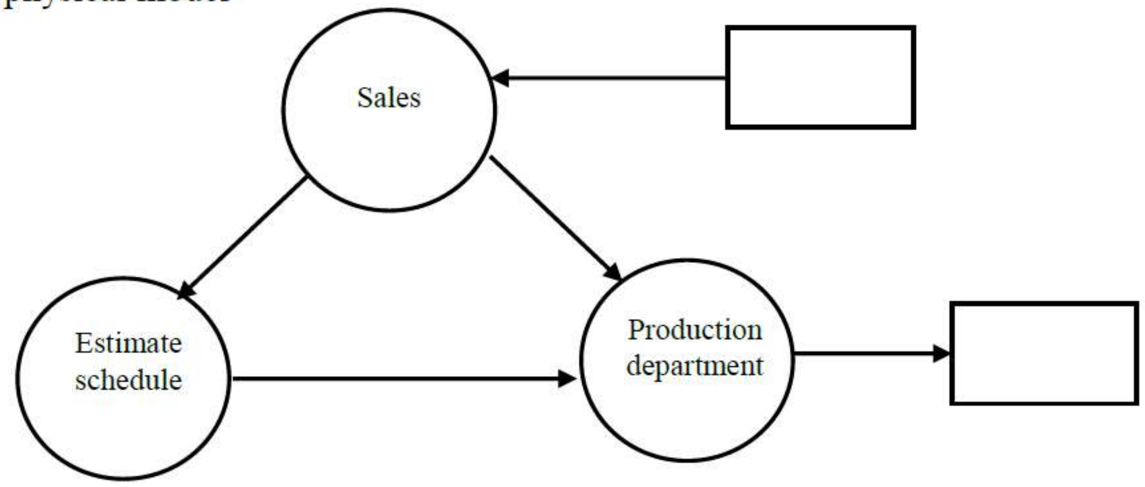
Physical design starts in step 4. Many activities take place during physical design. Decisions are made on what processes are to remain manual and which are to be computerized. Physical devices to store any data are chosen, and methods for carrying out system functions defined. The interface between system user and computers in the new system is designed.

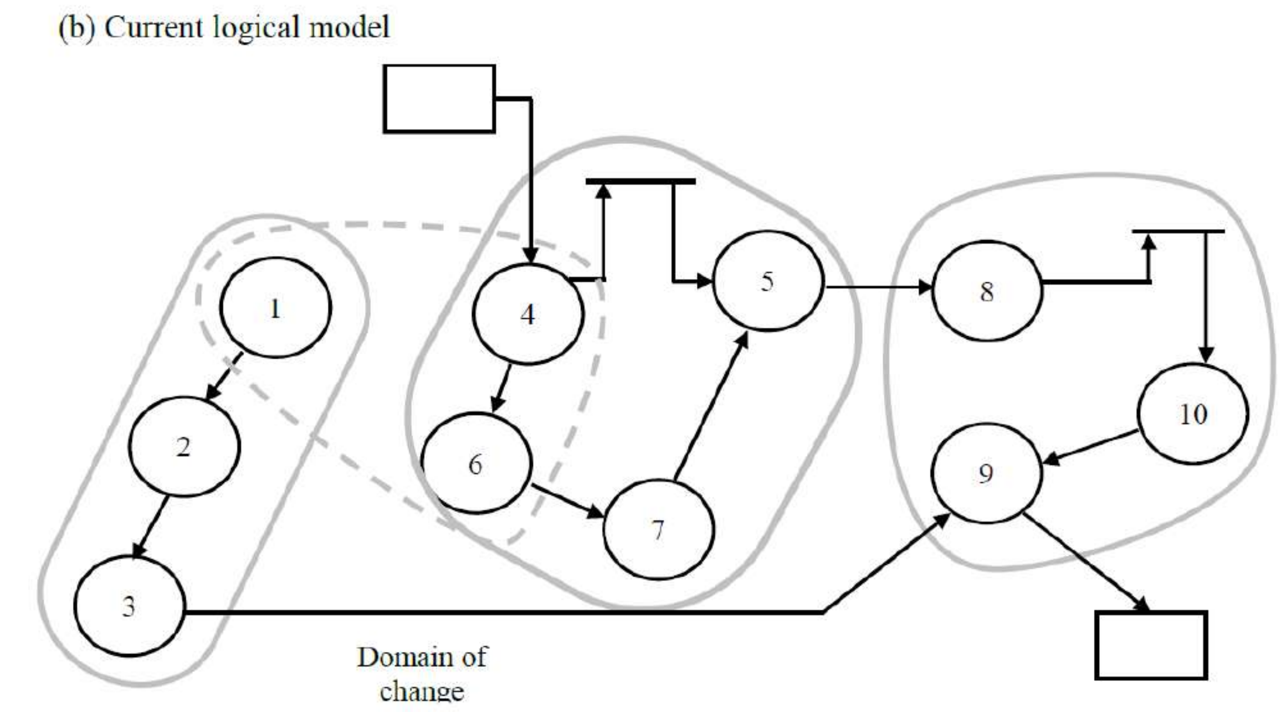
Note that four structured analysis descriptions are developed during the cycle shown in Figure 28. As shown in Figure 11.2 they are:

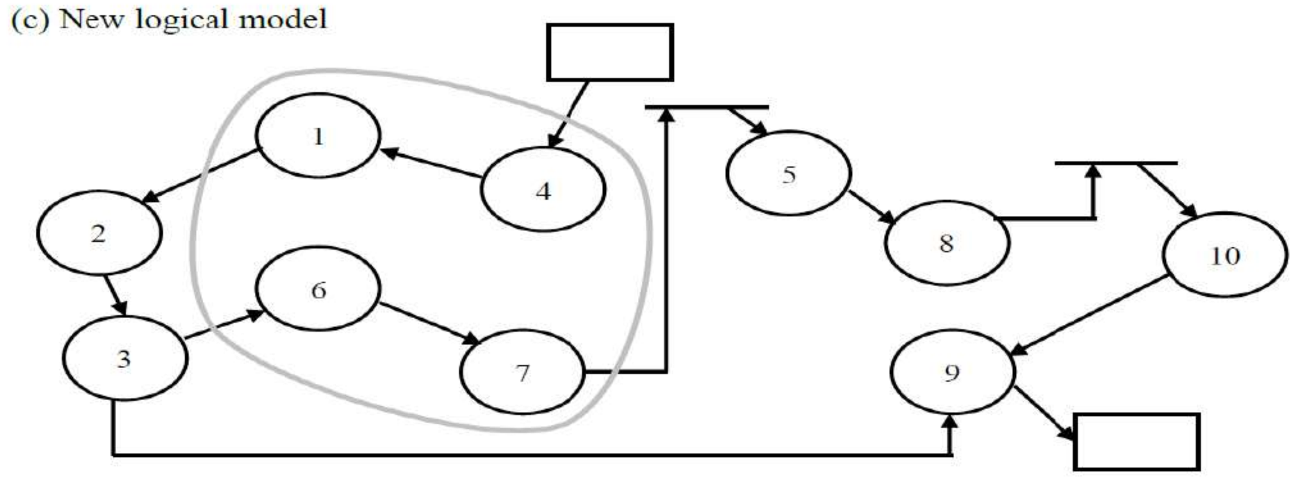
- a current physical model.
- a current logical model.
- a new logical model.
- a new physical model.

These problem-solving steps are combined with the system *life cycle*. Top-level logical and physical models are developed in Phase 4A. These models are elaborated into detailed physical design during Phase 4B of the system development cycle.

(a) Current physical model







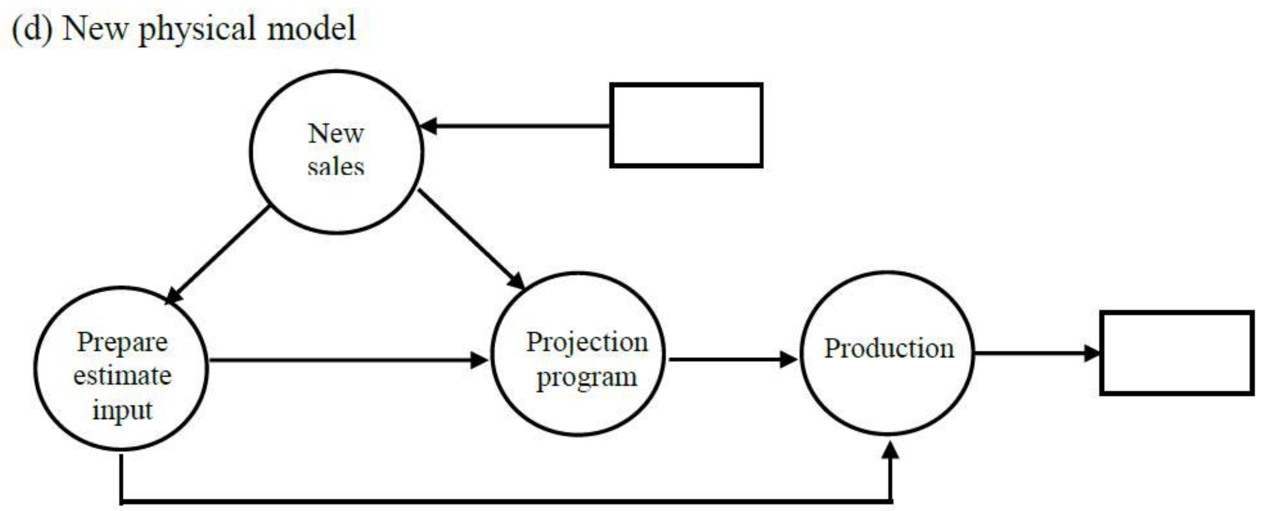


Figure 28: Problem-solving cycle

Designing the New Logical Model

Figure 29 shows a method for creating a new model from the current logical model of the existing system. The first step is to see which processes are affected by the objectives for the new system. These processes are known as the domain of change. The domain of change looks something like Figure 29. It includes all the processes that are affected by the objectives, together with the interface between these processes and the remainder of the system. The affected processes may be adjacent to each other as shown in Figure 29(a) or they may be made up of disconnected or disjointed data flow diagrams, as in Figure 29(b).

Redesigning the Domain of Change

A technique often used to redesign systems is illustrated in Figure 11.5. It starts with the domain of change and data flows in and out of this domain. The design then proceeds in the following steps:

1. Add data stores – Identify the data needed inside the domain of change and construct data stores for this data.
2. Define processes, data flows and data stores – Take each input into the domain of change and define processes that the input data flows through before it is stored or used to produce an output.
3. Add processes that transform data – Define processes that use data in the data stores and define how the processes use the data.
4. Add processes to create outputs – Look at each output produced and identify the data stores used to produce the output. Define the processes that the data goes through before being output.

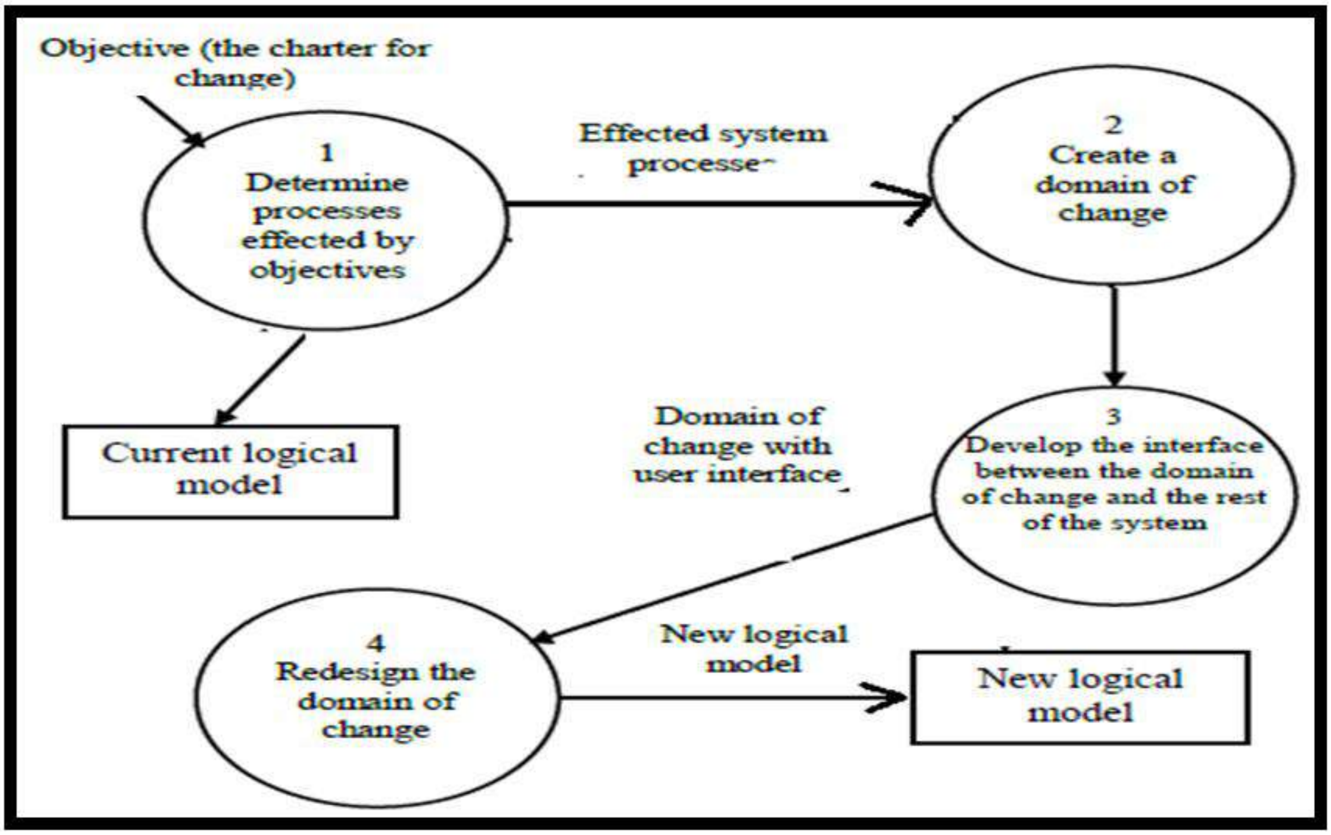

Objective (the charter for change)
1 Determine processes effected by objectives
Effected system processe
2 Create a domain of change
Current logical model
Domain of change with user interface
3 Develop the interface between the domain of change and the rest of the system
4 Redesign the domain of change
New logical model
New logical model

Figure 28: Steps for developing the new logical model

Of course, it may not be possible to proceed in this way for a large system. A top-down approach may be preferred instead. Figure 30 illustrates how top-level processes are defined. A list of major system functions is first made and their inputs and outputs defined. These functions are then linked by data flows to construct a top- level data flow diagram (DFD) for the domain of change. The method shown in Figure 30 is now applied to each top-level function.

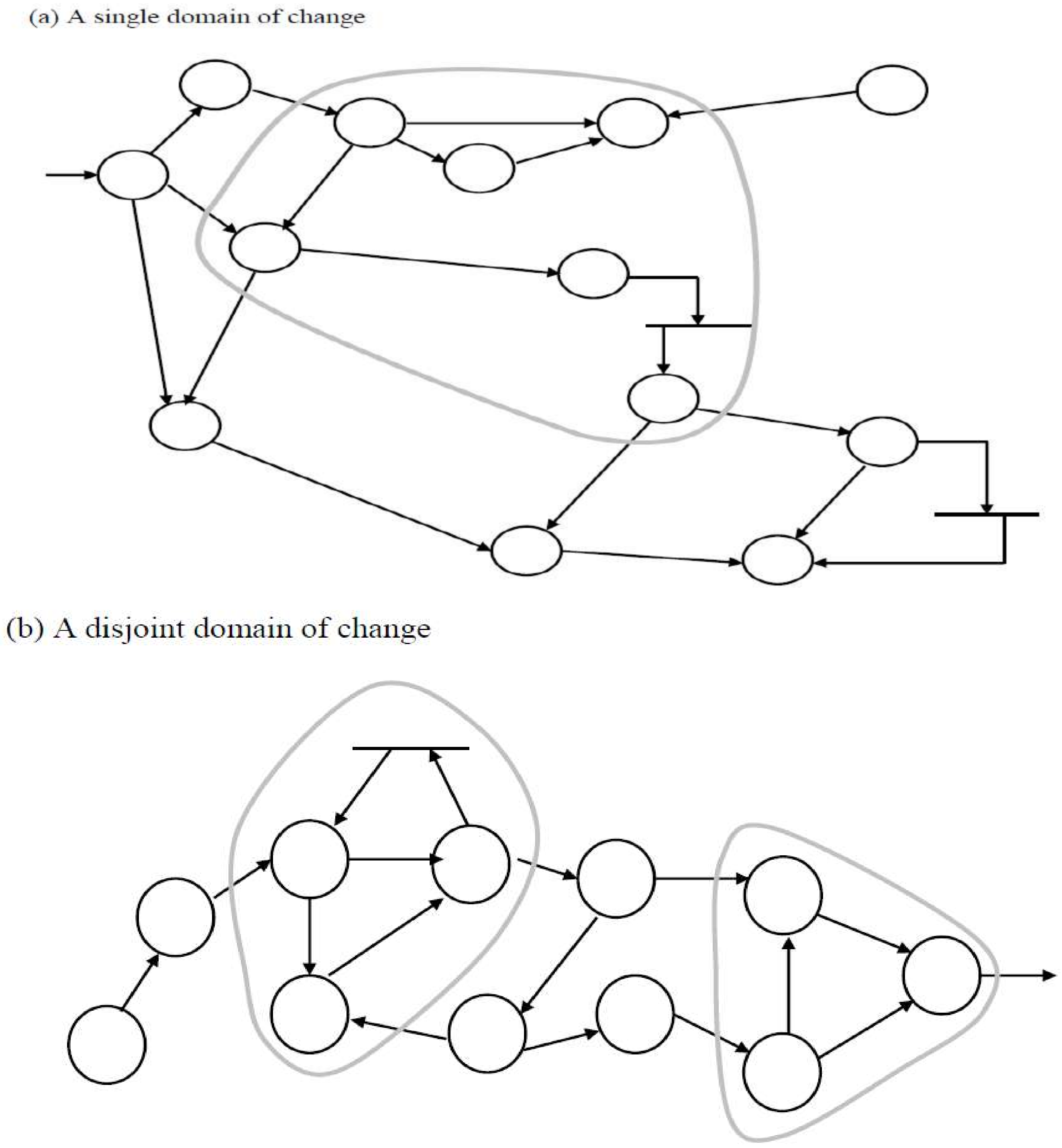

(a) A single domain of change
(b) A disjoint domain of change

Figure 29: Domain of change

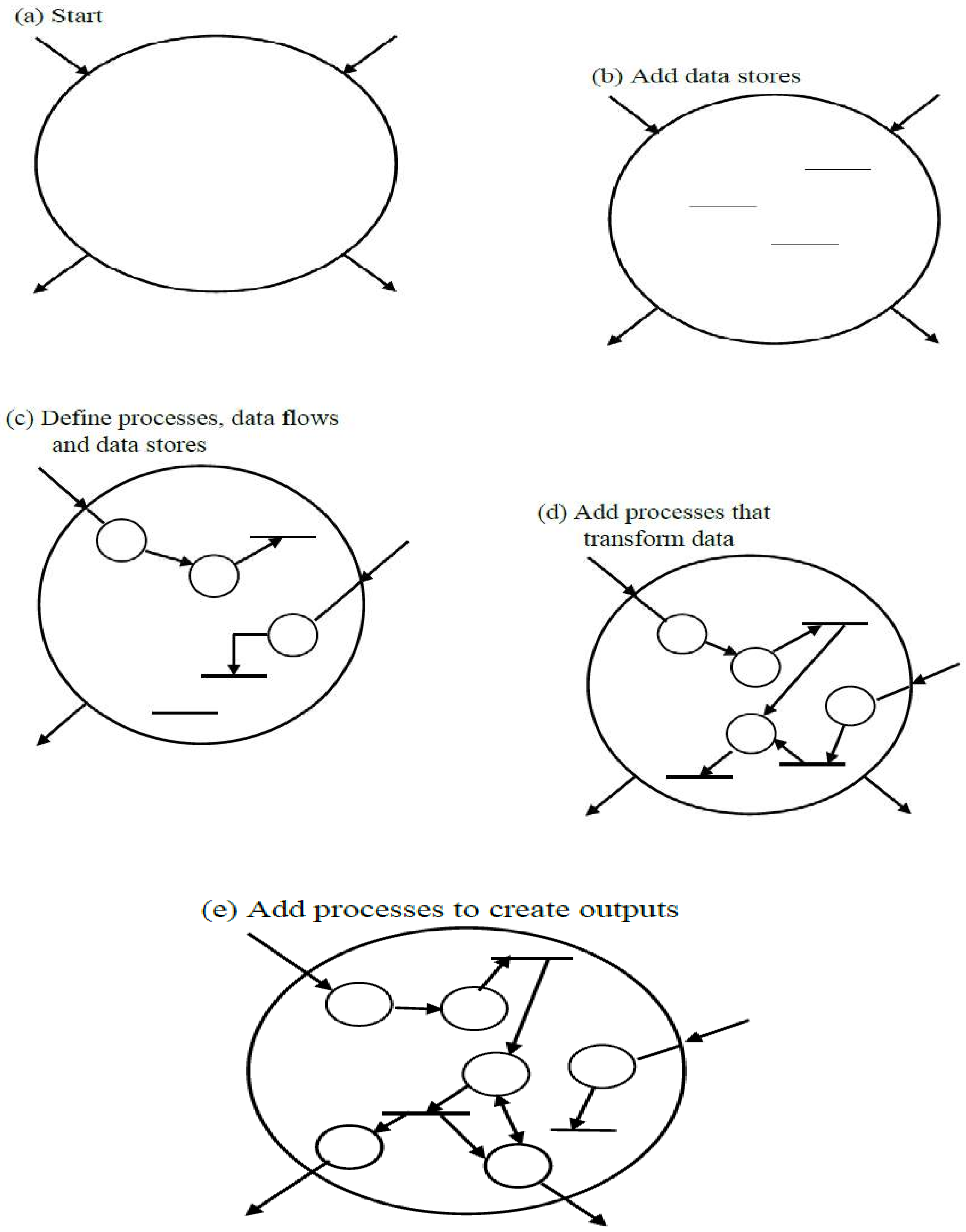

(a) Start
(b) Add data stores
(c) Define processes, data flows and data stores
(d) Add processes that transform data
(e) Add processes to create outputs

Figure 30: A set of steps used to design the new system

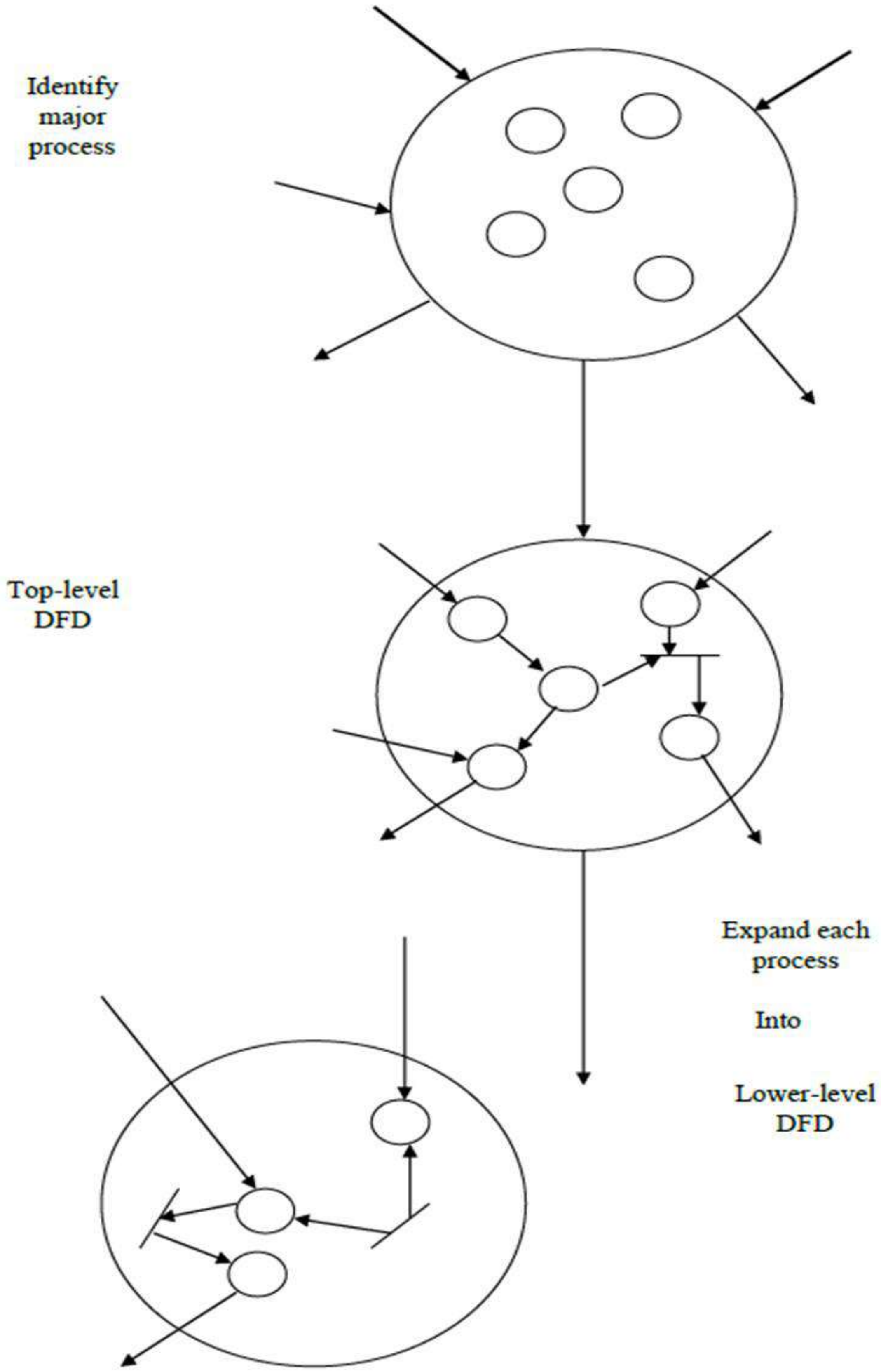


Figure 31: Top-down design

Data Analysis in Design

Data analysis follows the following steps during design:

1. Update the E-R level model to include any new system items.
2. Add the definitions of any new data elements to the data dictionary.
3. Normalize the model.
4. Specify frequency and usage requirements of the data items.

The newly-designed logical model should then be discussed with system users to gain their approval.

Amending an Existing System

The alternative to a complete redesign of the domain of change is to make changes to individual components of the domain of change – or redesign by parts.

A number of different kinds of amendments can be made to an existing system. Some amendments are:

- adding a new system process;
- creating a new data flow;
- changing the sequence of operations on information;
- eliminating redundant or unnecessary processes;
- combining two or more processes;
- adding new data and changing processes to use this data.

New Logical to New Physical

Physical design proceeds in two steps:

- an initial broad level design; followed by
- a detailed design

The broad physical design phase is made during Phase 4A of the linear cycle. Detailed design is completed in Phase 4B.

Broad-level physical design investigates alternate automation possibilities. It does this by drawing a boundary to identify the processes to be automated. This boundary is represented by a dashed line and is often called the *man-machine interface*. The man-machine interface can be drawn on high- and low-level data flow diagram. An example is shown in Figures 32 and 33. The leveled diagram shows that the transaction is entered in two parts. Part 1 is entered first. This is examined for errors. A request is made for Part 2 of the transaction if no errors are found in Part 1. Otherwise, a request is made for a correction.

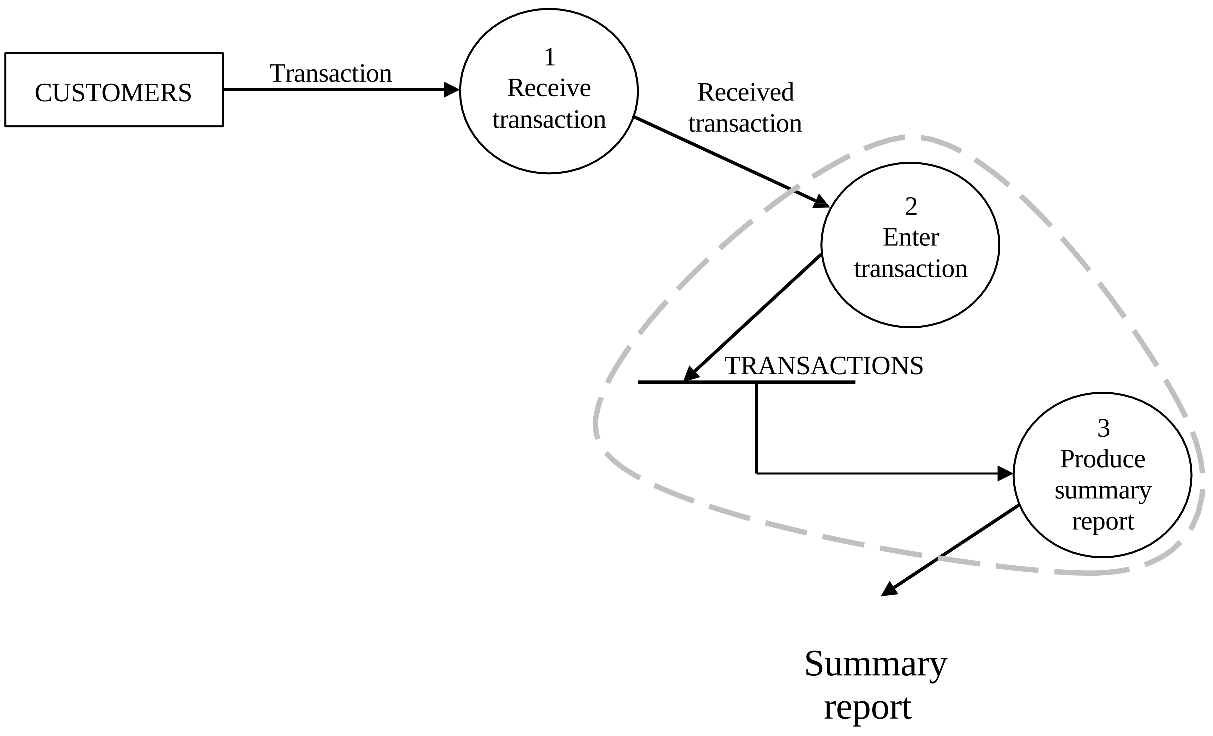


Figure 32: High-level man-machine boundary

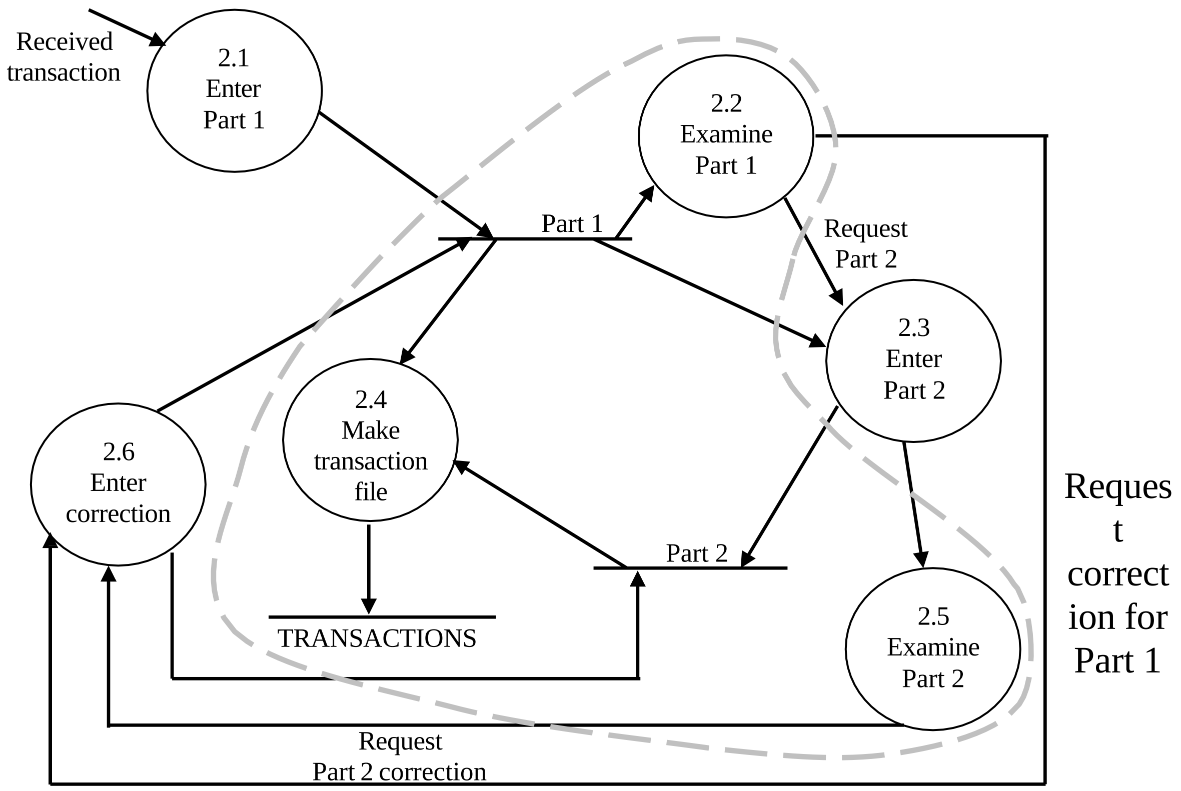


Figure 32: Low-level man-machine boundary

Detailed system design

Designing User Procedures

User procedures define what people must do to make the system work. Two things are important in user procedure design:

- Functional design, which is, making sure the system does what it is supposed to do. Functional design defines the tasks needed to make the system work.
- Job design, specifies what people must do within the tasks.

Functional Design

User procedures specify the tasks needed to capture data and pass it correctly through the system. The procedures must specify many things for each task. A user procedure must specify:

- how data is obtained,
- the medium used to carry the data, and
- the format of the data on the medium.

It must also specify the methods used to move the data and how to perform any calculation on the data.

A number of methods are used to design procedure tasks. A design method must satisfy a number of requirements. One requirement is to make sure that all the processes in the DFD are included in the user procedures. The simplest way to satisfy this requirement is to make each process in the DFD into task. This is assigned to one person or a group of people.

Describing User Procedures

User procedures define flows of information and the tasks needed to realize these flows. They also describe physical devices that are used to carry and store the data. User procedures are often described by flowcharts to represent physical system operation.

A flowcharting technique uses a finite set of symbols with each symbol representing a physical system component. One typical set of symbols is illustrated in Figure 12.1, which shows symbols used to represent computer components. Figure 12.2, on the other hand, illustrates symbols used to represent manual system components.

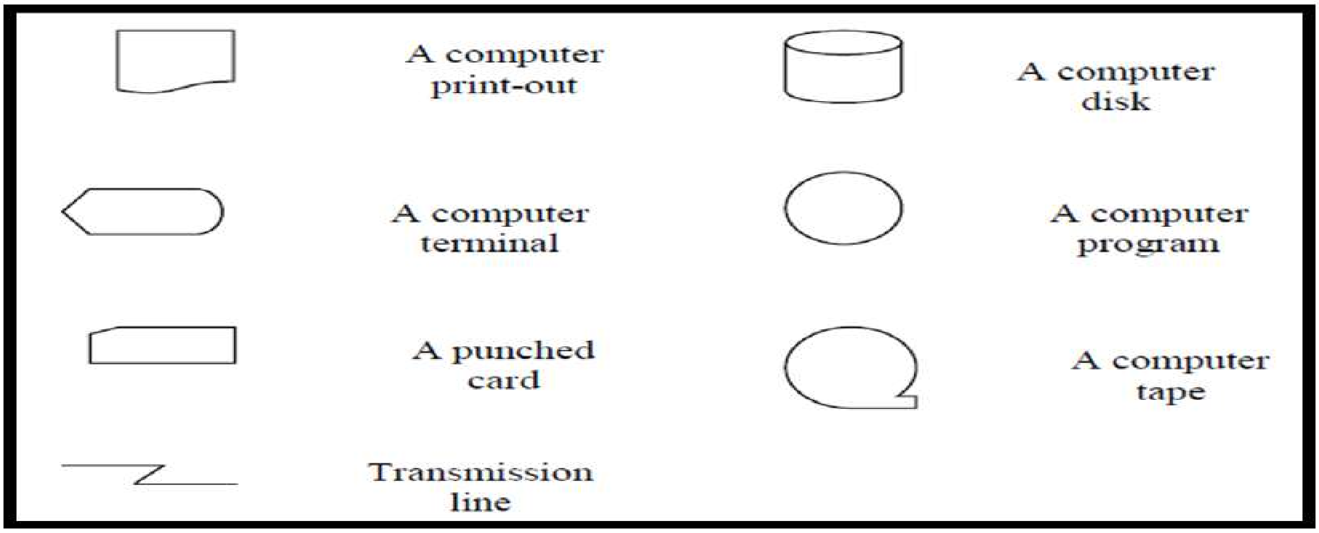


Figure 33: Flowcharting symbols for computer components

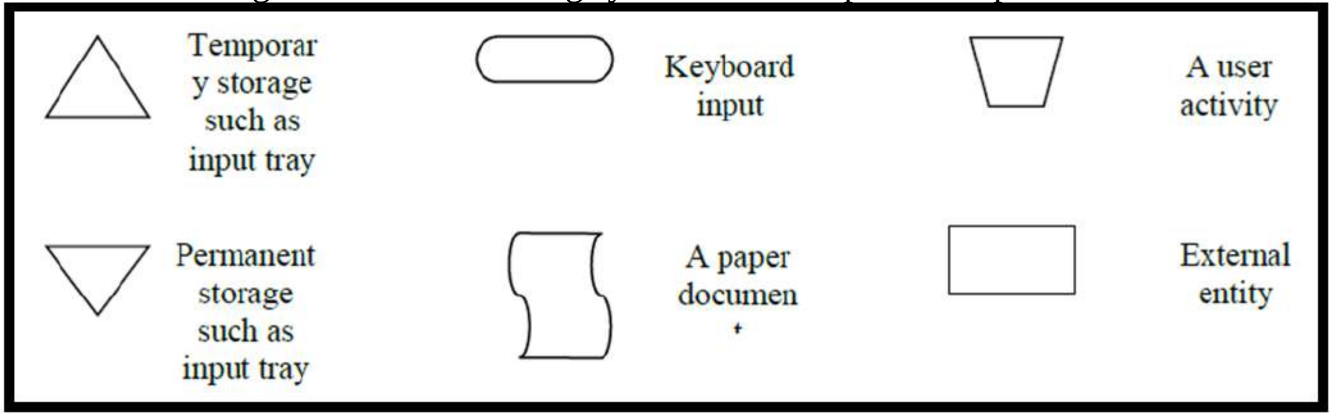

Temporar y storage such as input tray
Keyboard input
A user activity
Permanent storage such as input tray
A paper documen t
External entity

Figure 34: Flowcharting symbols for non-computer components

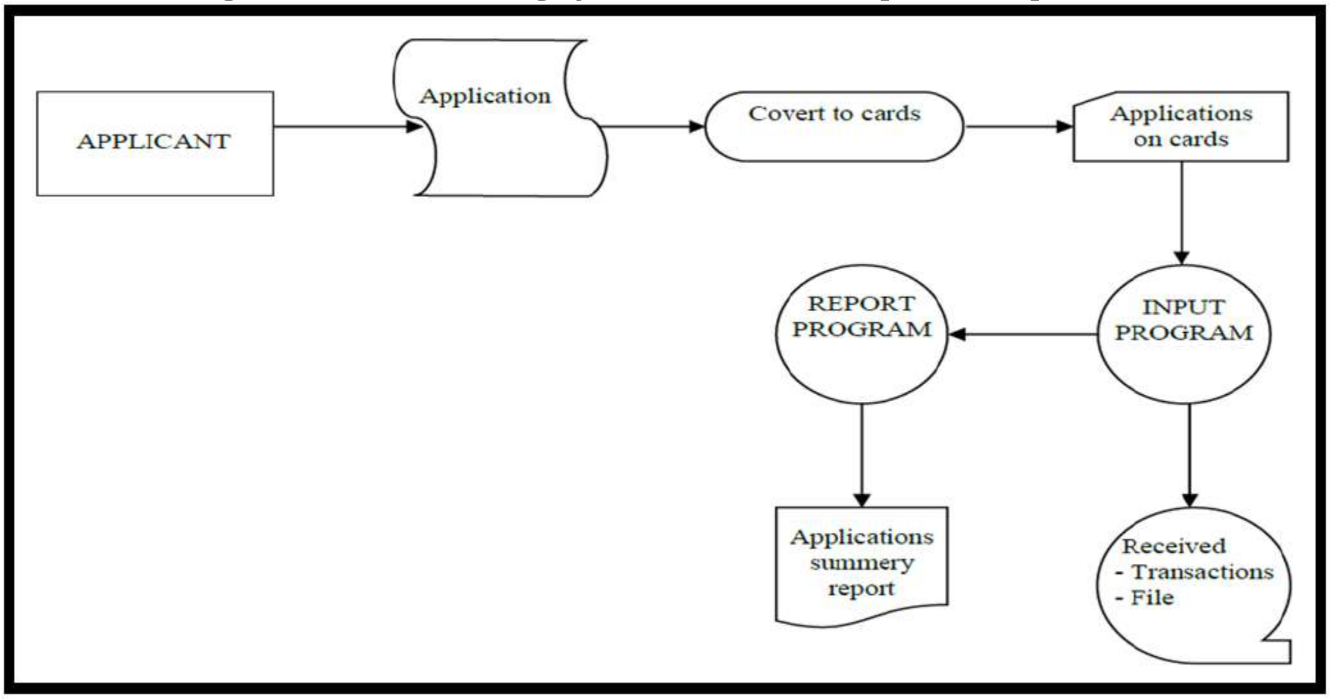


Figure 35: illustrates a typical flowchart.

Figure 35 – A flowchart, illustrates how applications are collected and input into computer. It combines manual and computer processes.

Once tasks in the user procedures are defined, it is necessary to specify what people must do to realize to tasks. This process is called *job design*. A job must be defined for all tasks in the system.

Job Design

Good job design involves a number of things. One is to ensure that *people* are not in the service of computers. This can be done in a number of ways. First, the interface to the computer should be made as easy and pleasant as possible. The interface should use dialog that approaches natural language rather than using computer jargon. Designing jobs with such variety often called *job enrichment*.

Another important consideration in job design is the skills and skill levels required for jobs. Skills include dealing with people, various technical skills and decision or supervision abilities.

Job descriptions are usually documented in a user manual. This manual is produced in conjunction with users during system design. It includes an entry for each task and the jobs necessary to accomplish that task.

Job Enrichment

Job enrichment, now a common term in practice, is used to make jobs more interesting and rewarding to system users. Job enrichment gives greater initiative and responsibility to individuals, enabling people in the organization to improve themselves through involvement in a greater variety of activities. They also get greater satisfaction from their work because they are contributing to the development of the organization rather than doing same minor and repetitive task.

An important part of job enrichment is to eliminate jobs made up of routine and repetitive tasks. Jobs can also be enriched by providing a good interface with the computer, ensuring that people do not feel they are in the service of computers rather than the reverse.

The Computer Interface

The purpose of the computer interface is quite simple. It is to capture information about the system from users and make information available to these or other users. Information capture is called the system input, which stores in database. The supply of information is called the system output.

Off-Line Processing

Off-line interfaces are designed to match off-line processing requirements. Off-line processing is often done in what is known as *batch mode*. A typical batch run is shown in Figure 36.

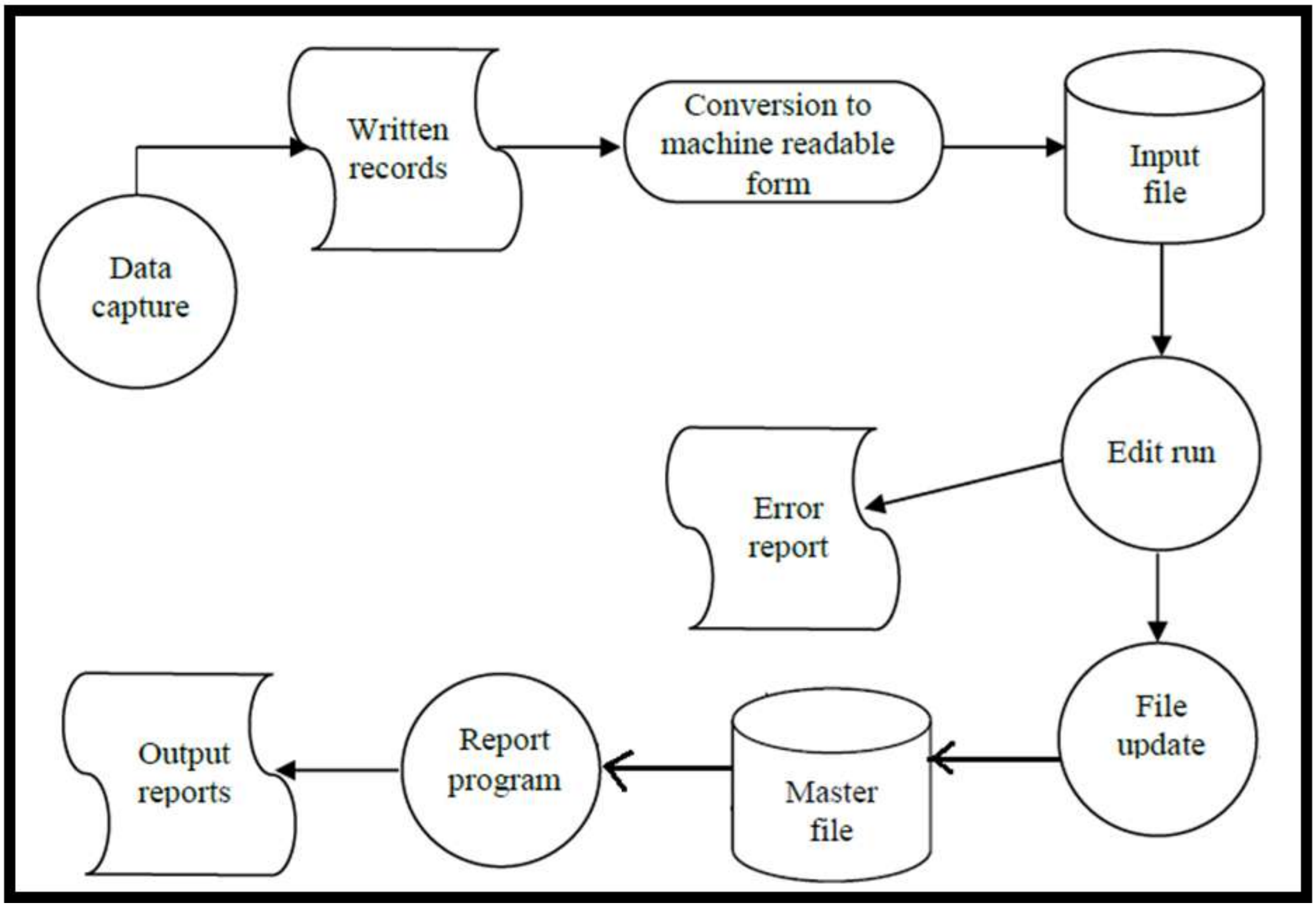


Figure 36: A batch run

Off-Line Input Interface

The most important input interface in off-line processing is the *form* that is used to capture the transactions. Form design is quite important for many reasons:

- forms must be easy to fill in.
- form layout must be clear.
- sometimes forms allow codes to be used.

On-Line Processing

On-line processing takes place through an interchange of messages between the user and computer in a relatively short time. This interchange of messages is called the *user dialog*.

Database design

Database design converts the data model developed in logical design to database definition that is supported by database software known as database management system (DBMS).

Database design proceeds through a number of steps, illustrated in Figure 13.1. The first step is DBMS independent, in this step the conceptual E-R or relational model to a set of record types known as *logical record structure* (LRS), with each record made up of a number of *fields*. The first step also defines how data in these records is to be used. These access requirements are used later to choose DBMS structures.

The next database steps convert the LRS to a database definition. These steps use techniques that depend on the DBMS. DBMS-dependent techniques are needed here because different DBMS support different kinds of links between their records.

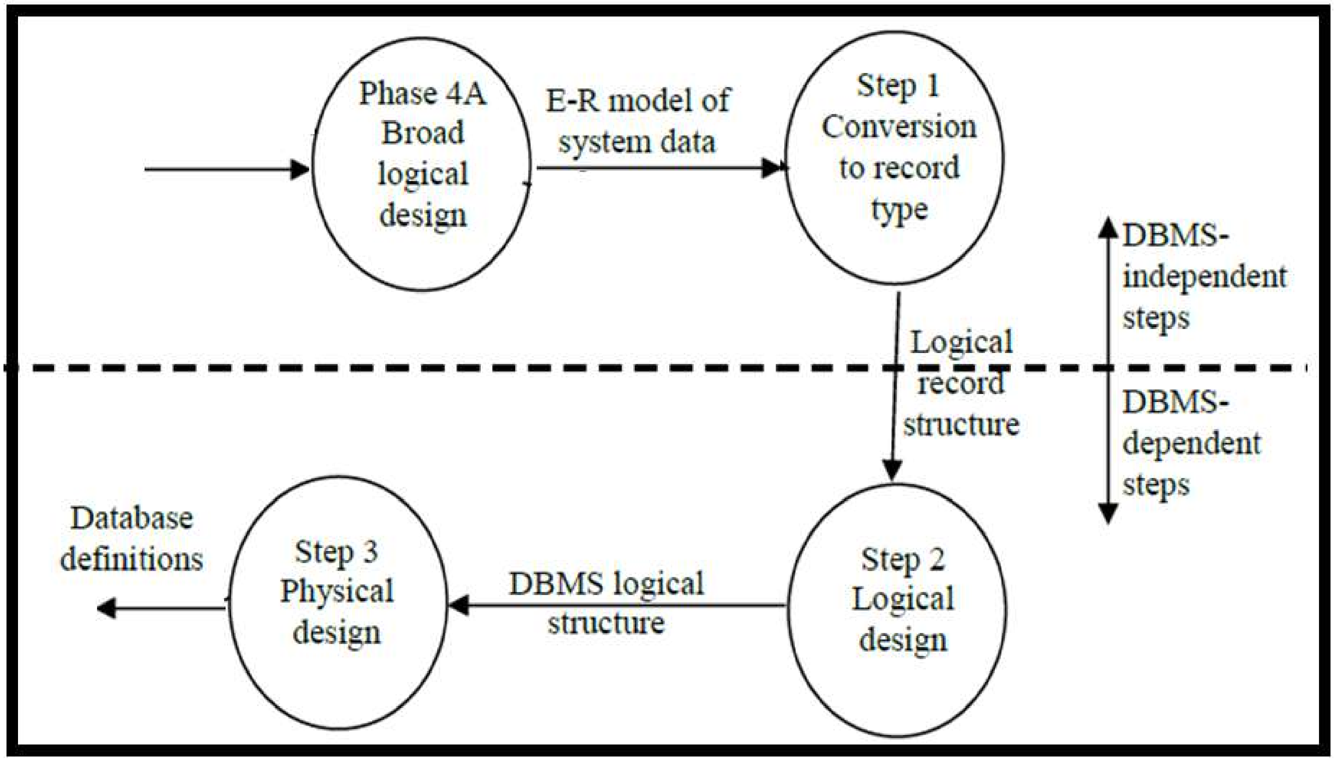


Figure 37: Database design steps

DBMS-dependent design proceeds in two steps. The first step is logical design. This step defines the DBMS record types and the links between them. The next step is physical design, which chooses a physical organization that supports the methods used to access the database.

Conversion to Logical Record Structures (LRS)

To drive LRS from the conceptual model, there are two methods can be used. One starts with a relational model of data. The other method starts with an E-R model and converts it directly to an LRS.

Converting the Relational Model to A LRS

The relational model of system data is converted to the LRS in two steps. Record types are created in the first step and links between the record types are added in the second step. In the first step, each relation is converted to a structure record type and each relation attribute to a record field.

Links between these record types are defined in the next step. To do this, it is necessary to use foreign keys. A set of attributes, F, is a foreign key in a relation, R, if:

1. F is not a relation key of R; but
2. F is a relation key of some other relation, say R2.

The next step is to create links between records in a LRS. There is a link from relation R1 to relation R2, if the relation key of R1 is a foreign key in R2. Consider the following relations:

PROJECTS (PROJECT-NO, START-DATE, BUDGET);
USE (PROJECT-NO, PART-NO, QTY-USED);
PARTS (PART-NO, DESC, PRICE);

Thus, the three relations will be converted to the LRS shown in Figure 13.2.

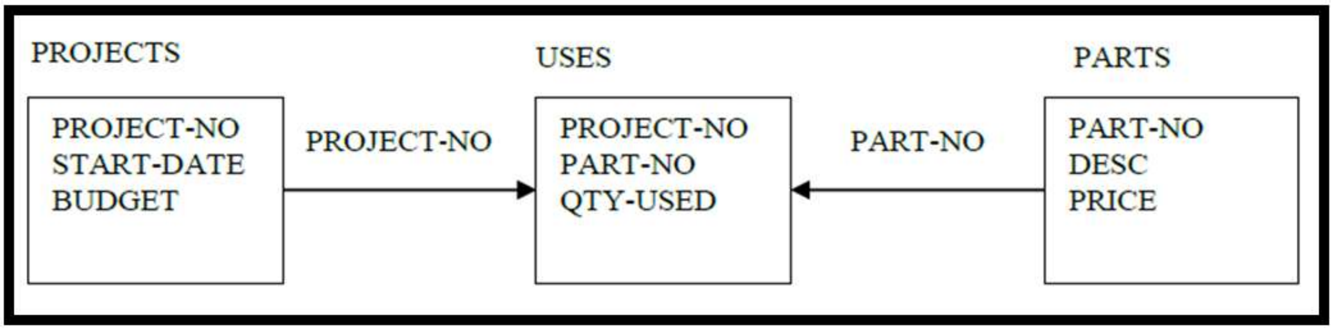


Figure 38: Logical Record Structure

Links are directed from one record type to another. Links also have another property. If you look at the link direction and the label on the link, you will find that the link originates on the record type that contains only one record with a given value of the link label.

Converting an E-R Model to A LRS

The simplest conversion is to make each set of the E-R diagram into a record type. However, there is one small variation to this where we combine object sets that have the same relation key to reduce the number of record types. One such conversion is illustrated in Figure 13.3. You will find that such combinations are always possible in 1: N relationships. In a 1: N relationship, entities in one of the entities sets appear in one relationship only.

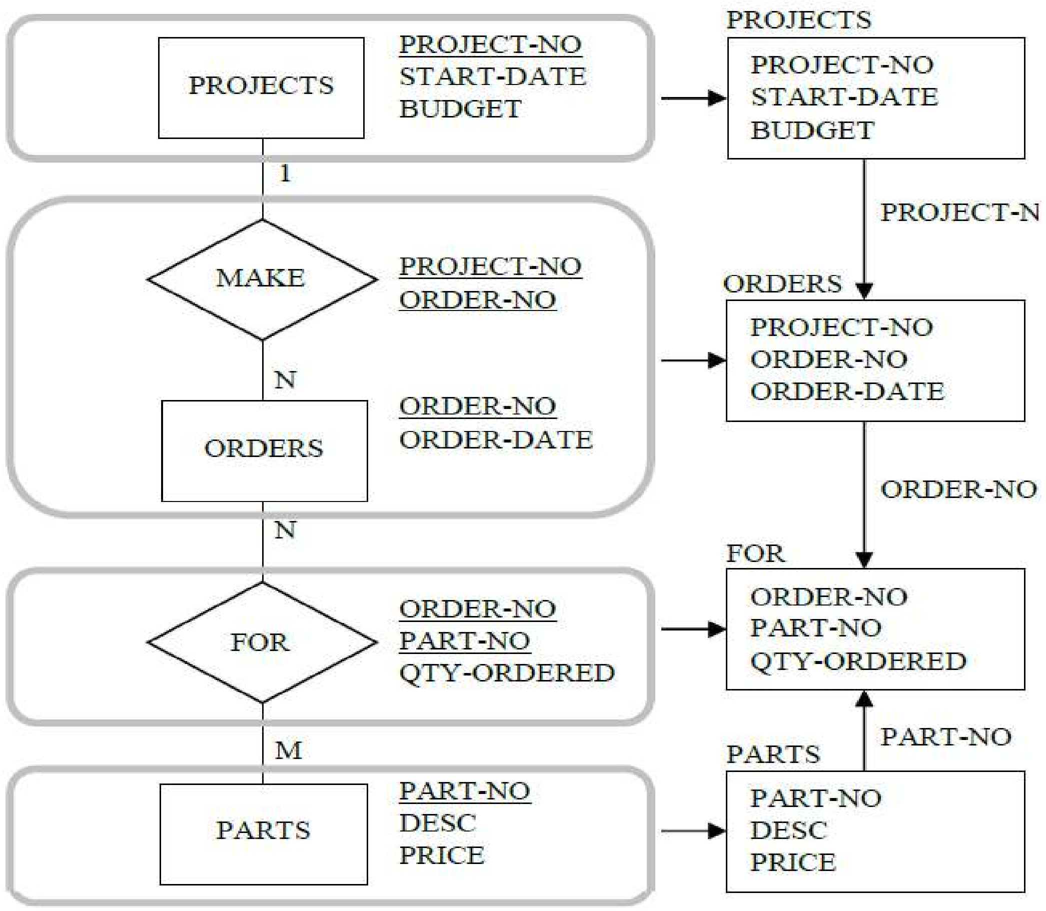


Figure 39: Converting an E-R diagram to a LRS

Links are then added. Links start on logical record type that represent entities and terminate on logical records that represent relationships. The same rule applies where sets have been combined, the link is from the record type, which represents an entity set, to a record type, which contains the relationship.

Completing the Database Specification

The logical record structure is just one part of the database specification. It serves to define the database structure. For database design, we need additional information, in particular:

- quantitative data, which tells us the volume of information to be stored;
- access requirements, which tell as how the database is to be used. Access requirements are used to choose file keys during database design.

Information about quantitative data and access requirements is gathered during systems analysis. Quantitative data consists of:

- the size of data items; and
- the number of assurances of record types.

The size of data items is usually given in a data is usually given in a data dictionary. Record volumes can be added to the LRS, by the method shown in Figure 13.5.

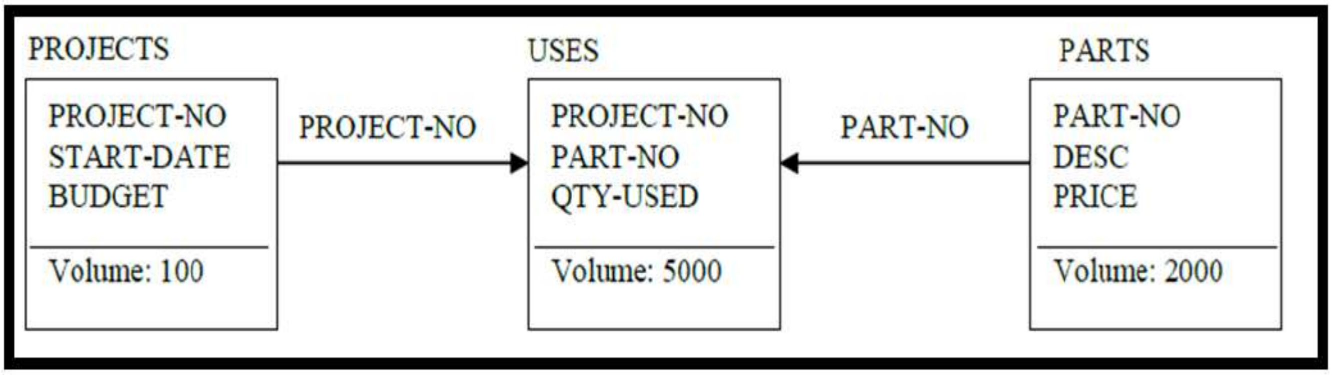


Figure 40: Add the number of records into LRS

Specifying Access Requirements

Access requirements are initially picked up from user procedure specifications, which include statements about how users will access data. These statements become the database access requirements. During database design, access paths are plotted against logical record types for each access requirement. The access paths show how data is to be used and describe:

- the record types accessed by each access request;
- the sequence in which the record types are accessed;
- the access keys used to select record types;
- the items retrieved from each record; and
- the number of records accessed.

There are many ways to draw access paths. One method is illustrated in Figure 13.6. This figure shows three access requirements plotted against three logical record types. The three access requirements are:

- A – find the START-DATE for given project.
- B – find the PRICE of parts used on a project with a given PROJECT-NO.
- C – find the START-DATE of projects that use a given PART-NO.

There is one access path for each access requirement and each access path can be made up of or more access steps. Each access step labeled with an access key. The access step also includes a description of the activity at the step.

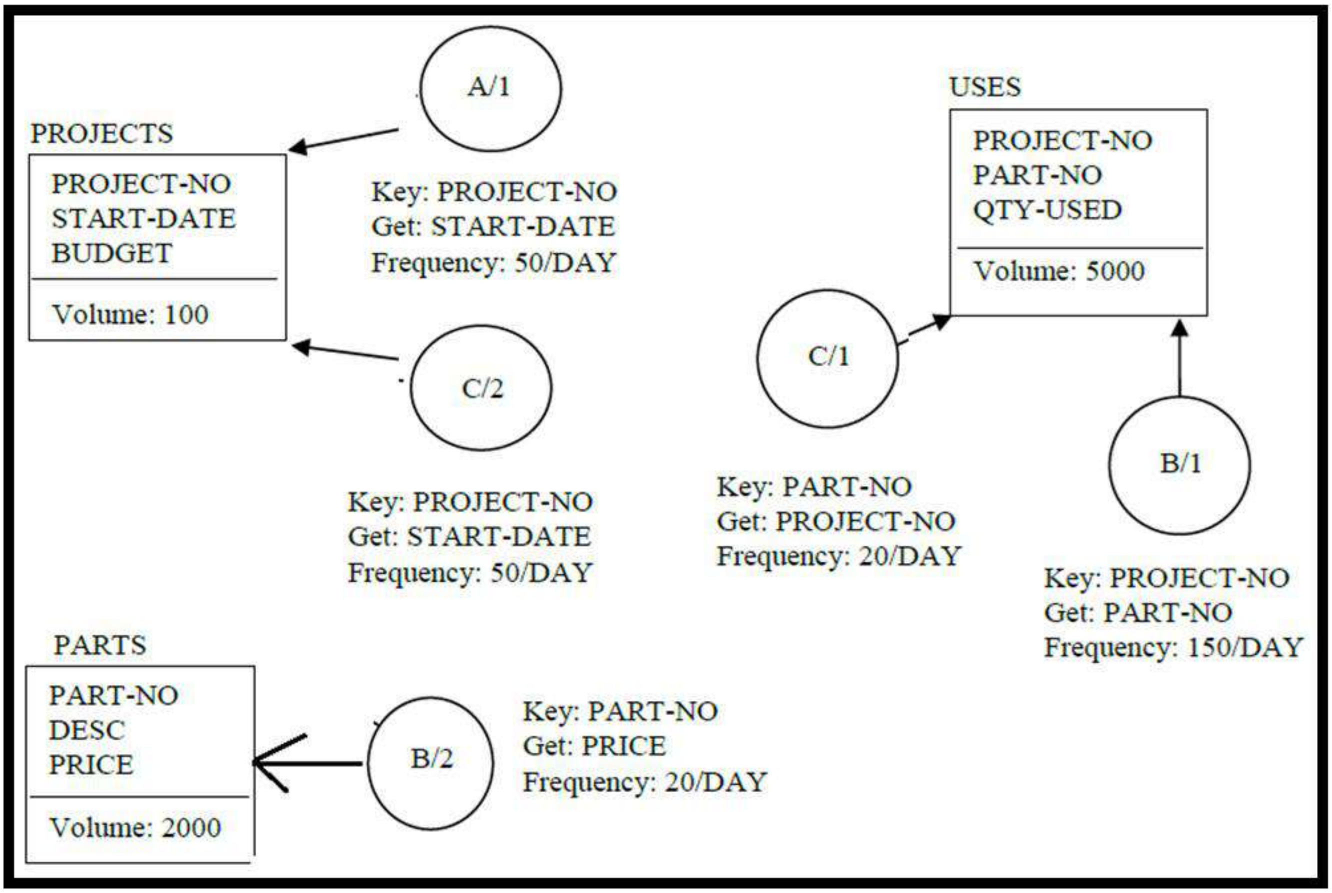


Figure 41: Access paths to the logical structure level

Defining the access requirements completes the database specification, which is now used to start database design. The logical record structure is converted to a database logical design. Access paths are used to select appropriate physical structure.

Conversion to A Set of Files

The first step is to converts each logical record to a file. Thus, the LRS in Figure 13.6 would be converted to three files. The logical record field would become the file fields.

The keys in the access steps are used to choose keys for file indexes. All access to the file can be direct. Figure 13.7 illustrates the file structure that is obtained from the LRS of Figure 13.6. The file structure is made up of tree files with indexes. The fields of records in each file are shown next to the file. You should also note that links between record types in LRS are not carried over into the file structure because file software does not support links between records. Database management software, however, does support such links.

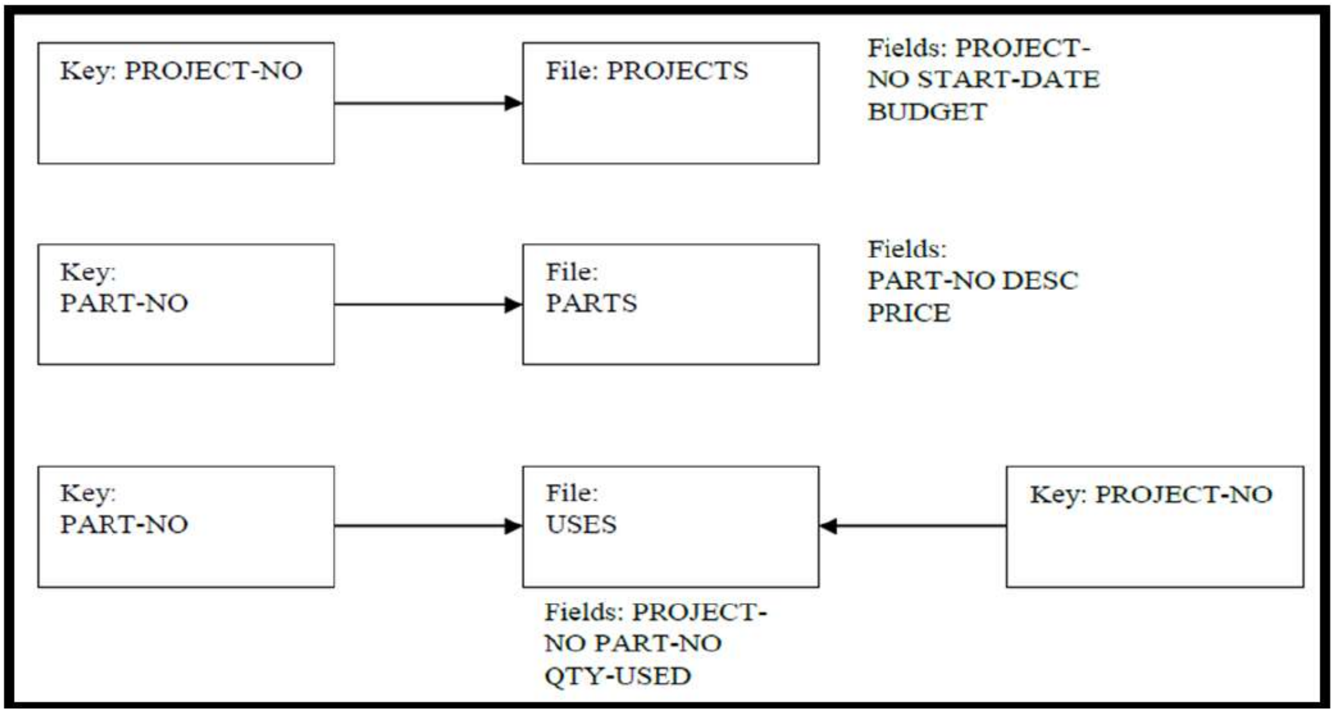

Key: PROJECT-NO
File: PROJECTS
Fields: PROJECT-NO START-DATE BUDGET
Key: PART-NO
File: PARTS
Fields: PART-NO DESC PRICE
Key: PART-NO
File: USES
Key: PROJECT-NO
Fields: PROJECT-NO PART-NO QTY-USED

Figure 42: A file structure for the logical structure

Program Design

Program design is the last component of detailed design. The program specification produced at the end of program design must ensure that programs satisfy user requirements as defined by user procedures.

Apart from satisfying user requirements, programs must satisfy a number of other criteria. First, programs must be easy to read and understand. Second, programs should accommodate system changes that always accrue after a system is built. Program should be easy to maintain to support such changes. Finally, the third requirement is that programs should be written to use computer resources in the most effective manner.

These objectives are achieved by *modular program design* and *structured programming*. Modular program design localizes each well-defined user system function to one program module. Structured programming uses block-structured constructs to make programs readable. Note that modularity and structured programming are consistent with the criteria for good DFD. In a good DFD each process represents one well-defined function. Furthermore, each process is described using structured English, which employs constructs similar to structured programming. These DFD structure to the program specification.

Program design uses the other two parts of detailed design, user design and database design. Program design uses user procedures to define the computer interface. It uses the database design to write I/O code, which accesses those database parts needed by the program.

Steps In Program Design

The steps in program design and their relationships with other parts of detailed design are illustrated in Figure 43. Program design starts with DFD inside the automation boundary and uses outputs from database design and user procedure design to produce working programs.

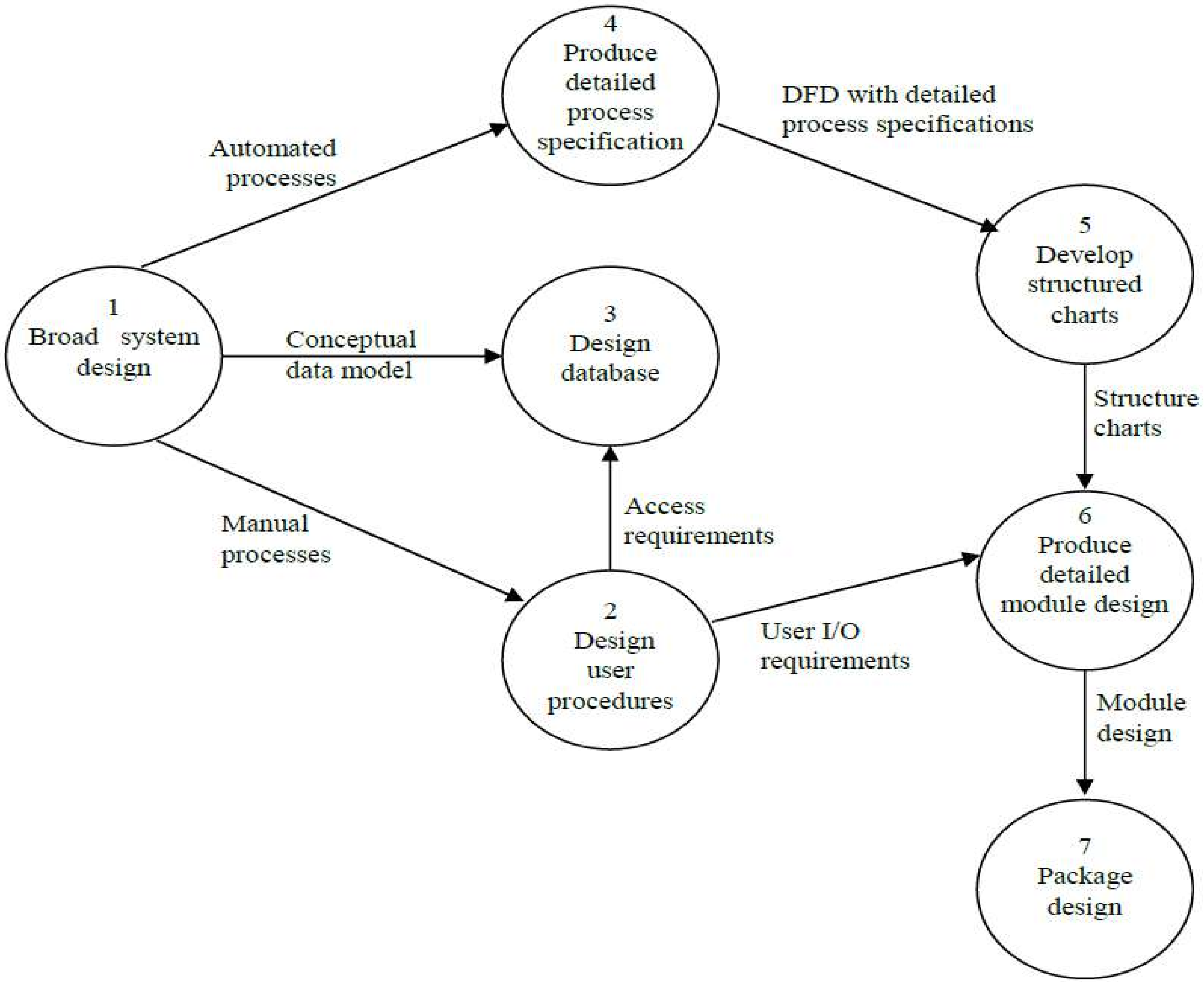
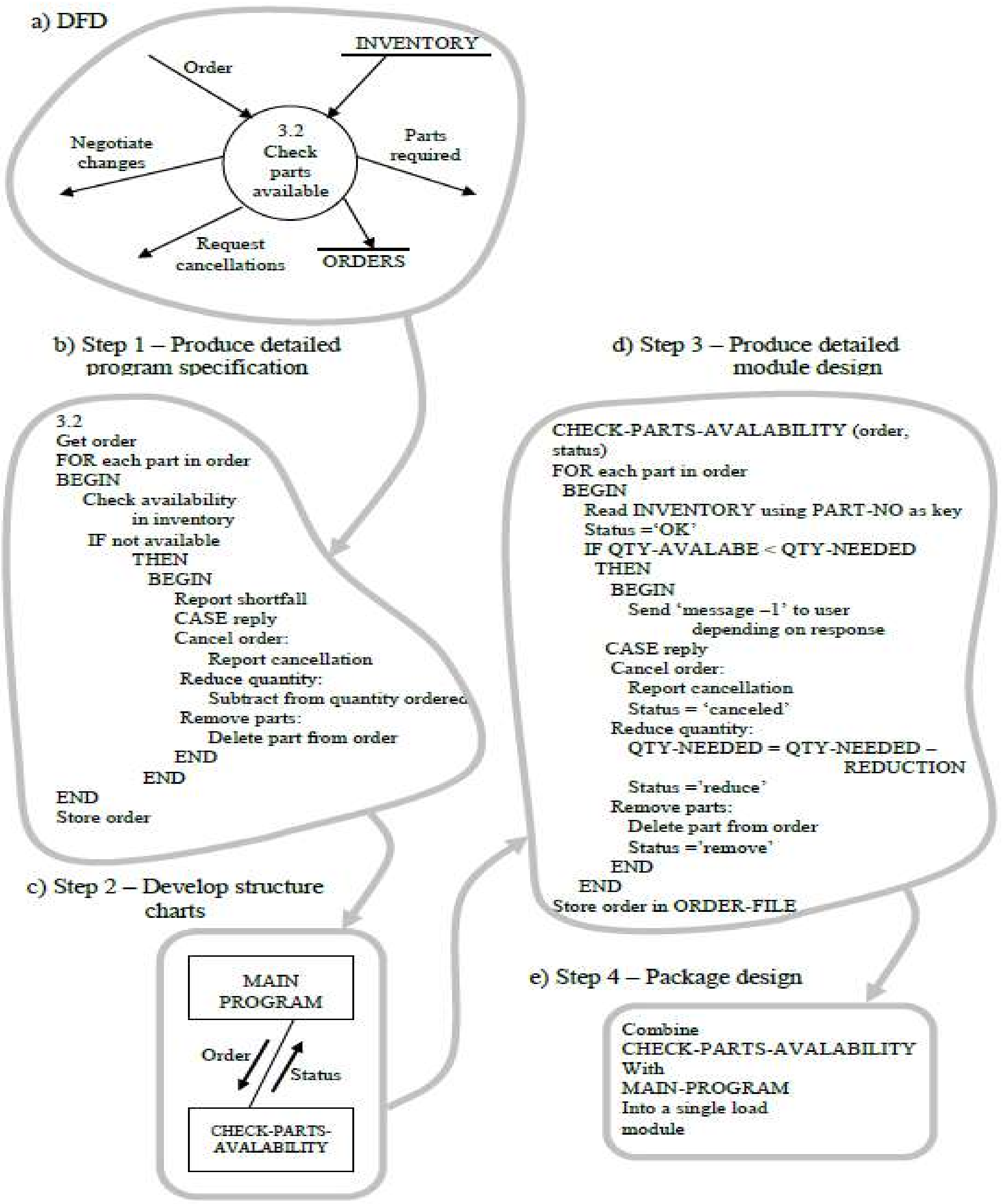


Figure 43: Program Design

Transition from DFD to Detailed Program Specifications

Figure 44 – describe the program design steps in more detail. It uses Process 3.2 to illustrate these steps. The first step develops detailed process specification for the process. Detailed process specifications are written in Structured English and begin to approximate program code. They include all the conditions that can arise in the process and how they are treated.

The next step the DFD are converted to structure charts and each process in the DFD become a program module in the structure charts. The structure charts show the program modules and how they pass information – as parameters –between themselves.


a) DFD
INVENTORY
Order
3.2 Check parts available
Negotiate changes
Parts required
Request cancellations
ORDERS
b) Step 1 – Produce detailed program specification
c) Step 2 – Develop structure charts
MAIN PROGRAM
Order
Status
CHECK-PARTS-AVALABILITY
d) Step 3 – Produce detailed module design
e) Step 4 – Package design

```
3.2
Get order
FOR each part in order
BEGIN
    Check availability
          in inventory
    IF not available
          THEN
            BEGIN
                Report shortfall
                CASE reply
                Cancel order:
                     Report cancellation
                 Reduce quantity:
                     Subtract from quantity ordered
                 Remove parts:
                     Delete part from order
                END
            END
END
Store order
```

```
CHECK-PARTS-AVALABILITY (order,
status)
FOR each part in order
  BEGIN
     Read INVENTORY using PART-NO as key
     Status ='OK'
     IF QTY-AVALABE < QTY-NEEDED
       THEN
         BEGIN
           Send 'message –1' to user
                     depending on response
         CASE reply
          Cancel order:
            Report cancellation
            Status = 'canceled'
          Reduce quantity:
            QTY-NEEDED = QTY-NEEDED –
                                     REDUCTION
            Status ='reduce'
          Remove parts:
            Delete part from order
            Status ='remove'
          END
     END
Store order in ORDER-FILE
```

```
Combine
CHECK-PARTS-AVALABILITY
With
MAIN-PROGRAM
Into a single load
module
```

Figure 44: Program design steps

Once we know how program modules call each other, we can convert detailed process specifications to detailed module designs. As shown by Step 3 in Figure 44, that detailed module designs include procedure calls and parameters passed to a module. They also include user dialogs, error checks, I/O code and references to the database. They use database design and user procedure design to choose the correct database references and user dialog. Finally, the modules are packaged into load modules. Load modules are groups of program modules that are loaded into the memory together.