

Contents

1	Chapter 1: INTRODUCTION TO SYSTEMS	1
1.1	Overview.....	1
1.1.1	Definition of System.....	1
1.1.1.1	Common types of systems	2
1.1.1.2	Natural systems.....	2
1.1.1.3	Man-made systems	4
1.1.1.4	Automated systems.....	5
1.1.1.5	General systems principles	6
1.1.2	Participants to system development.....	6
1.1.2.1	Users	7
1.1.2.2	Management	10
1.1.2.3	Auditors	10
1.1.2.4	Systems analysts	10
1.1.2.5	Systems designers.....	11
1.1.2.6	Programmers.....	11
1.1.2.7	Operations personnel	11
1.1.3	System development Life-cycle	12
1.1.3.1	Initial investigation	12
1.1.3.2	Feasibility Study	12
1.1.3.3	General Design	12
1.1.3.4	Detailed Design	12
1.1.3.5	Implementation	13
1.1.3.6	System Audit	13
1.2	System approach.....	13
1.2.1	System requirements.....	14
1.2.1.1	The system's requirements:.....	14
1.2.1.2	Users requirements:	15
1.2.1.3	Technical requirements:.....	15

1.2.2	System survey	16
1.2.3	Investigation Methods	18
1.2.3.1	Survey methods:	18
1.2.3.2	Observation methods:	19
1.2.3.3	Questionnaire method:	19
1.2.4	Survey report	20
1.2.5	Function analysis	23
2	Chapter 2: SYSTEMS ANALYSIS	25
2.1	Analysis of structured system	25
2.1.1	Overview of system analysis	25
2.1.2	Nature of analysis	26
2.1.3	Importance of analysis in system's life cycle	27
2.1.4	Role and requirement of system analysts	28
2.1.5	Popular methods of system analysis	28
2.1.6	Tools supporting analysis: Functioning diagram, data flow diagram and relationship diagram	29
2.1.6.1	Functional diagram (FD):	29
2.1.6.2	Data flow diagram: describe the information flow in the system	31
2.1.6.3	Entity relationship diagram (ERD)	40
2.2	Information analysis, data model	45
2.2.1	Requirements of data approach	45
2.2.2	Data model and its main components	45
2.2.3	Data model development	47
2.2.3.1	Defining entity types	47
2.2.3.2	Identifying relationship	48
2.2.4	The advanced data modeling	48
2.2.4.1	Optional relationship	48
2.2.4.2	Abstract entity	48
2.2.4.3	Recursive relationship	49

2.3	Strengthening of information structure - relationship model	49
2.3.1	Identifying attributes.....	50
2.3.2	Determining types of entities (normalization).....	55
2.3.3	Defining relationships.....	63
2.3.4	Models comparison.....	64
2.4	Completion of information analysis	64
2.4.1	Corporation of new requirements	65
2.4.2	Supporting tools and materials	65
2.4.2.1	Process Description	66
2.4.2.2	Dictionary	66
2.4.2.3	Seminar	67
2.4.3	Summary of analysis process.....	68
 Chapter 3: SYSTEM DESIGN		 71
3.1.	The overall designing specifications	71
3.1.1	Overview of structured design.....	71
3.1.2.	Design goals and objectives.....	73
3.1.3.	Relationship between system analysis and design.....	74
3.1.4.	Principles of procedure design.....	75
3.1.5.	The stages of design.....	76
3.1.6	Tools and Techniques of design	77
3.2.	Structured Design as process	85
3.2.1	The phases of design.....	85
3.2.2	Identification of the computer system	89
3.2.3.	Model Construction	92
3.3.	User-computer interface design.....	92
3.3.1.	Dialogue design	92
3.3.2	Screen design	94
3.3.3.	Outputs Design	94

3.4. System monitoring design	103
3.5. Organizing the system's components	104
3.5.1 Grouping criteria.....	104
3.5.2 Designing approach	105
3.5.3 Modelization tools and techniques	106
3.6. Analysis of data usage and logical navigation.....	108
3.6.1 Path analysis diagram	109
3.6.2 Navigation model.....	109
3.6.3 Data usage schema.....	110
3.7. Design of Databases	111
3.7.1 The common problem of database design	111
3.7.2 An ideal database structure	117
3.7.3 Physical database design.....	118
3.7.4 Designing process	120
3.7.5 Physical storage structure design.....	120
References.....	129
Index	131

1 Chapter 1: INTRODUCTION TO SYSTEMS

In this chapter, the material highlights a number of concepts of systems, the workload and reasons behind the workload in a system's life cycle. In this part, you'll get to know the techniques applied in studying survey methods, topic discussions, specification of systems' requirement and analysis skills. After reading this part, you'll gain more experience in dealing with systems issues which managers, users of different levels, and technicians, consider through different views.

1.1 Overview

1.1.1 Definition of System

There are many common types of systems that we come into contact with every day. It is important to be familiar with different kinds of systems for at least two reasons:

- First, even though your work as a systems analyst will probably focus on one kind of system – an automated, computerized information system – it will generally be a part of a larger system. For example, you may be working on a payroll system, which is part of a larger “human resources” system, which is, in turn, part of an overall business organization (which is itself, a system), which is, in turn, part of a larger economic system, and so on. Thus, to make your system successful, you must understand the other systems with which it will interact. Many of the computer systems that we build are replacements, or new implementations of, non-computerized systems that are already in existence. Also, most computer systems interact with, or interface with, a variety of existing systems (some of which may be computerized and some which may not). If our new computer system is to be successful, we must understand, in reasonable detail, how the current system behaves.
- Second, even though many types of systems appear to be quite different, they turn out to have many similarities. There are common principles and philosophies and theories that apply remarkably well to virtually all kinds of systems. Thus, we can often apply to systems that we build in the computer field, what we have learned about other systems, based on our own day-to-day experience, as well as the experience of scientists and engineers in a variety of fields.

Thus, if we understand something of general systems theory, it can help us better understand computerized (automated) information systems. Today, this is more and more important, because we want to build stable, reliable systems that will function well in our complex society, and of course, there are many examples of non-computer systems that have survived for thousands of years.

And now, we can consider a definition of the basic term "system". It provides several definitions:

1. A regularly interacting or interdependent group of items forming a unified whole.
2. An organized set of doctrines, ideas, or principles, usually intended to explain the arrangements or working of a systematic whole.
3. An organized or established procedure.
4. Harmonious arrangement or pattern: order.
5. An organized society or social situation regarded as stultifying establishment.

1.1.1.1 Common types of systems

There are many different types of systems, but indeed, virtually everything that we come into contact with during our day-to-day life is either a system or a component of a system (both).

It is useful to organize the many different kinds of systems into useful categories. Because our ultimate focus is on computer systems, we will divide all systems into two categories: natural systems and man-made systems.

1.1.1.2 Natural systems

There are a lot of systems that are not made by people: they exist in nature and, by and large, serve their own purpose. It is convenient to divide natural systems into two basic subcategories: physical systems and living systems.

Physical systems include such diverse example as:

- Stellar systems: galaxies, solar systems, and so on.
- Geological systems: rivers, mountain ranges, and so on.
- Molecular systems: complex organizations of atoms.

Physical systems are interesting to study because we sometimes want to modify them. We also develop a variety of man-made systems, including computer systems, which must interact harmoniously with physical systems; so it is often important to be able to model those systems to ensure that we understand them as fully as possible.

Living systems encompass all of the myriad animals and plants around us, as well as our own human race. The properties and characteristics of familiar living systems can be used to help illustrate and better understand man-made systems.

The living systems, whether at the level of the cell, the organ, the organism, the group, the organization, the society, or the supranational system, all contain the 19 subsystems:

- 1 . The reproducer, which is capable of giving rise to other systems similar to the one it is in.
- 2 . The boundary, which holds together the components that make up the system, protects them from environmental stresses, and excludes or permits entry to various sorts of matter-energy and information.
- 3 . The ingestor, which brings matter-energy across the system boundary from its environment.
- 4 . The distributor, which carries inputs from outside the system or outputs from its subsystems around the system to each component.
- 5 . The converter, which changes certain inputs to the system into forms more useful for the special processes of that particular system.
- 6 . The producer, which forms stable associations that endure for significant periods among matter-energy inputs to the system or outputs from its converter, the materials synthesized being or growth, damage repair, or replacement of components of the system, or for providing energy for moving or constituting the system's outputs of products or information markets to its suprasystem.
- 7 . The matter-energy storage subsystem, which retains in the system, for different periods of time, deposits of various sorts of matter-energy.
- 8 . The extruder, which transmits matter-energy out of the system in the form of products or wastes.
- 9 . The motor, which moves the system or parts of it in relation to part or all of its environment or moves components of its environment in relation to each other.
- 10 . The supporter, which maintains the proper spatial relationships among components of the systems, so that they can interact without weighing each other down or crowding each other.
- 11 . The input transducer, which brings markers bearing information into system, changing them to other matter-energy forms suitable for transmission within it.
- 12 . The internal transducer, which receives, from other subsystems or components within the system, markers bearing information about significant alterations in those subsystems or components, changing them to other matter-energy form of a sort that can be transmitted within it.

- 13 . The channel and net, which are composed of a single route in physical space, or multiple interconnected routes, by which markers bearing information are transmitted to all parts of the system.
- 14 . The decoder, who alters the code of information input to it through the input transducer or internal transducer into a private code that can be used internally by the system.
- 15 . The associator, which carries out the first stage of the learning process, forming enduring associations among items of information in the system.
- 16 . The memory, which carries out the second stage of the learning process, storing various sorts of information in the system for different periods of time.
- 17 . The decider, which receives information inputs from all other subsystems and transmits to them information outputs that control the entire system.
- 18 . The encoder, who alters the code of information input to it from other information processing subsystems, from a private code used internally by the system into a public code that can be interpreted by other systems in its environment.
- 19 . The output transducer, which puts out markers bearing information from the system, changing markers within the system into other matter-energy forms that can be transmitted over channels in the system's environment.

Keep in mind that many man-made systems (and automated systems) interact with living systems. In some cases, automated systems are being designed to replace living systems. And in other cases, researchers are considering living systems as components of automated systems.

1.1.1.3 Man-made systems

Man-made systems include such things as:

- 1 . Social systems: organizations of laws, doctrines, customs, and so on.
- 2 . An organized, disciplined collection of ideas.
- 3 . Transportation systems: networks of highways, canals, airlines and so on.
- 4 . Communication systems: telephone, telex, and so on.
- 5 . Manufacturing systems: factories, assembly lines, and so on.
- 6 . Financial systems: accounting, inventory, general ledger and so on.

Most of these systems include computers today. As a systems analyst, you will naturally assume that every system that you come in contact with should be computerized.

And the customer or user, with whom you interact will generally assume that you have such a bias. A systems analyst will analyze, or study, the system to determine its essence: and understand the system's required behavior, independent of the technology used to implement the system. In most case, we will be in a position to determine whether it makes sense to use a computer to carry out the functions of the system only after modeling its essential behavior.

Some information processing systems may not be automated because of these common reasons: Cost; Convenience; Security; Maintainability; Politics.

1.1.1.4 Automated systems

Automated systems are the man-made systems that interact with or are controlled by one or more computers. We can distinguish many different kinds of automated systems, but they all tend to have common components:

1. Computer hardware (CPUs, disks, terminals, and so on).
2. Computer software: system programs such as operating systems, database systems, and so on.
3. People: those who operate the system, those who provide its inputs and consume its outputs, and those who provide manual processing activities in a system.
4. Data: the information that the system remembers over a period of time.
5. Procedures: formal policies and instructions for operating the system.

One way of categorizing automated systems is by application. However, this turns out not to be terribly useful, for the techniques that we will discuss for analyzing, modeling, designing, and implementing automated systems are generally the same regardless of the application. A more useful categorization of automated systems is as follows:

1. Batch system: A batch system is one which in it, the information is usually retrieved on a sequential basis, which means that the computer system read through all the records in its database, processing and updating those records for which there is some activity.
2. On-line systems: An on-line systems is one which accepts input directly from the area where it is created. It is also a system in which the outputs, or results of computation, are returned directly to where they are required.
3. Real-time systems: A real-time system may be defined as one which controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time.
4. Decision-support systems: These computer systems do not make decisions on their own, but instead help managers and other professional “knowledge workers” in an

organization make intelligent, informed decisions about various aspects of the operation. Typically, the decision-support systems are passive in the sense that they do not operate on a regular basis: instead, they are used on an ad hoc basis, whenever needed.

5. Knowledge-based systems: The goal of computer scientists working in the field of artificial intelligence is to produce programs that imitate human performance in a wide variety of “intelligent” tasks. For some expert systems, that goal is close to being attained. For others, although we do not yet know how to construct programs that perform well on their own, we can begin to build programs that significantly assist people in their performance of a task.

1.1.1.5 General systems principles

There are a few general principles that are of particular interest to people building automated information systems. They include the following:

1. The more specialized a system is, the less able it is to adapt to different circumstances.
2. The more general-purpose a system is, the less optimized it is for any particular situation. But the more the system is optimized for a particular situation, the less adaptable it will be to new circumstances.
3. The larger a system is, the more of its resources that must be devoted to its everyday maintenance.
4. Systems are always part of larger systems, and they can always be partitioned into smaller systems.
5. Systems grow. This principle could not be true for all systems, but many of the systems with which we are familiar do grow, because we often fail to take it into account when we begin developing the system.

1.1.2 Participants to system development

In the role of a systems analyst, you will work on systems development projects with a variety of other people. The cast of characters will change from project to project, the personalities will differ dramatically, and the number of people that you interact with will range from as few as one to as many as several dozen. However, the roles are fairly consistent, and you will see them over and over again.

In a typical systems development project, there are the following major categories of players:

- User
- Management
- Auditors, quality assurance people, and ‘standards bearers’
- Systems analysts
- Systems designers
- Programmers
- Operations personnel

1.1.2.1 Users

The user, the most important player in the systems game, is the person (or group of people) for whom the system is being built. He or she is the person whom will be interviewed, often in great detail, to learn what features the new system must have to be successful. The user is the “owner” in the sense that he or she receives, or inherits-and thus owns- the system when it is finally built. And the user is the “customer” in at least two important respects:

- 1 . As in so many other professions, “the customer is always right”, regardless of how demanding, unpleasant, or irrational he or she may seem.
- 2 . The customer is ultimately the person paying for the system and usually has the right and/or the ability to refuse to pay if he or she is unhappy with the product received.

In most cases, it is fairly easy to identify the user: the user is typically the person who makes a formal request for a system. In a small organization, this is usually a very informal process. In a large organization, the initiation of a systems development project is usually much more formalized.

Whenever possible, the systems analyst should try to establish direct contact with the user. Even if other people are involved as intermediaries, it is important to have regular, face-to-face meeting with the person who will ultimately inherit the system.

If it is not possible to communicate directly with the user, then the documentation produced by the systems analyst becomes even more crucial.

The heterogeneity of Users

One mistake often made by computer programmers, and sometimes by systems analysts is to assume that all users are the same. “User” implies that the systems analyst will only have to interact with one person, even when it is obvious that more than one user is

involved, there is a tendency to think of them as a formless, shapeless, homogeneous group of humans. Here are two ways of categorizing users:

- Job category, or level of supervision.
- Level of experience with data processing.

Categorizing Users by Job category

On a typical systems analysis project, we can usually count on interacting with three main job categories:

- a. Operational users: are the clerical, operational, and administrative people most likely to have the most day-to-day contact with the new system. There are three things should be kept in mind when working with operational-level users:
 - Operational users are very much concerned with the functions that the system will perform, but they are likely to be even more concerned with the human interface issues.
 - Operational users tend to have a “local” view of the system, they tend to be knowledgeable about the specific job that they do and the people with whom they have immediate contact. But they often are unfamiliar with the “big picture”, that is, they may have trouble describing how their activity fits into the overall organization or what the overall organization’s charter really is.
 - Operational users tend to think of systems in very physical terms, that is, in terms of the implementation technology currently used to implement the system or in terms of technology that they imagine could be used.
- b. Supervisory users are employed in a supervisory capacity: they usually manage a group of operational users and are responsible for their performance. The significant things to remember about supervisory users are these:
 - Many of them are former operational users who have been promoted to their current position.
 - One reason that the supervisory user may be perceived as out of touch with the operational user is that he or she is often measured and motivated by performance against a budget.
 - It is usually the supervisory user who thinks of a new system as a way of reducing the number of operational users or avoiding further increases in their numbers as the volume of work increases.

- The supervisory user will often act as a middleman between the systems analyst and the operational user.
 - The supervisory user often thinks in the same physical terms as the operational user, and this perspective is often just as local as that of the operational user.
 - Finally, the supervisory user will be contacted day-to-day. He or she is the one who will typically define the requirements and detailed business policy that the system must implement. He or she may be a passive member of the team, a full-time member of the team, or even the project manager.
- c Executive-level users: are generally not directly involved in a systems development project, unless the project is so large and so important that it has a major impact on the organization.

To summarize, there are three different types, or levels, of users. Keep in mind that they have different perspectives, different interests and priorities, and different backgrounds. These three types of users can be characterized as shown below:

- Operational: Usually has local view. Carries out the function of the system. Has a physical view of the system.
- Supervisory user: May or may not have local view. Generally familiar with operation. Driven by budget considerations. Often acts as a middleman between users and higher levels of management.
- Executive user: Has a global view. Provides initiative for the project. No direct operating experience. Has a strategic concern.

Categorizing Users by Level of Experience

Different users will have different levels of experience. Today, we can distinguish between rank amateurs, cocky novices, and a small (but rapidly growing) number of true computer experts.

- The amateur is the one who has never seen a computer and who exclaims loudly and frequently that he or she doesn't understand all this computer stuff. The real problem with the amateur user is somewhat more subtle: he or she may find it difficult to understand the "language" that the systems analyst uses to describe the features, functions and characteristics of the system to be built, even though that language avoids obvious computer-related terminology.
- The second type of user is the "cocky novice", the person who has been involved in one or two systems development projects, or the user who has a personal computer and who has written one or two basic programs. This user often claims to know exactly what he

or she wants the system to do and is prone to point out all the mistakes that the systems analyst made on the last project.

- Of course, there are some users who really understand systems analysis, as well as the underlying technology of computers. It is a pleasure working with these people. The only problem may be that the user and the systems analyst derive so much pleasure talking about the tools and techniques of systems analysis that they forget that their true objective is to build a functioning system.

1.1.2.2 Management

Management is a rather loose term. The systems analyst is likely to come into contact with several different kinds of managers:

- User managers: managers in charge of several people in the operational area where the new system will be used. These are usually middle-level managers who want systems that will produce a variety of internal reports and short-term trend analyses.
- Executive development project (EDP)/MIS managers: the person in charge of the systems development project itself, and the higher-level managers who are concerned with the overall management and allocation of resources of all the technical staff in the systems development organization.
- General management: top-level managers who are not directly involved in the EDP organization or in the user organization. This might include the president and/ or chairman of the organization.

The primary interaction between the systems analyst and all these managers has to do with the resources that will be assigned to the project. It is the systems analyst's job to identify and document the user's requirements and the constraints within which the system must be built.

1.1.2.3 Auditors

Depending on the size of the project and the nature of the organization you work in, you may or may not have auditors.

1.1.2.4 Systems analysts

The system analyst is a key member of any systems development project. In a boarder sense, the systems analyst plays several roles:

- Archaeologist and scribe: As a systems analyst, one of the main jobs is to uncover detail and to document business policy that may exist only as “tribal folklore”, passed down from generation to generation of users.
- Innovator: The systems analyst must separate the symptoms of the user’s problem from the true causes. With his or her knowledge of computer technology, the analyst must help the user explore useful, new applications of computers.
- Mediator: The systems analyst who often finds himself in the middle of users, managers, programmers, auditors, and various other players, all of whom frequently disagree with one another.
- Project leader: Because the systems analyst is usually more experienced than the programmers on the project, and since he is assigned to the project before the programmers begin working, there is a natural tendency to assign project management responsibilities to the analyst.

1.1.2.5 Systems designers

The systems designer is the person (or group of people) who will receive the output of the systems analysis work. His or her job is to transform a technology-free statement of user requirements into a high-level architectural design that will provide the framework within which the programmer can work. In many cases, the systems analyst and the systems designer are the same person, or member of the same unified group of people. It is important for the systems analyst and systems designer to stay in close touch throughout the project.

1.1.2.6 Programmers

Particularly on large systems development projects, the systems designers are likely to be a “buffer” between the systems analysts and the programmers. The systems analysts deliver their product to the system designers, and the system designers deliver their product to the programmer. There is another reason why the systems analyst and the programmer may have little or no contact with each other: work is often performed in a strictly serial sequence in many systems development projects. Thus, the work of systems analysis takes place first and is completely finished before the work of programming begins.

1.1.2.7 Operations personnel

The operations personnel who are responsible for the computer center, telecommunications network, security of the computer hardware and data, as well as the actual running of computer programs, mounting of disk packs, and handling of output

from computer printers. All this happens after a new system has not only been analyzed and designed, but has also been programmed and tested.

1.1.3 System development Life-cycle

The six phases in the system development life cycle can be identified by different names. Also, there are no definite rules regarding what must be included in each of the six phases.

1.1.3.1 Initial investigation

This phase introduces the objectives of the initial investigation, the steps required to initiate an investigation; the tasks involved in the initial investigation, and the data gathering and interviewing techniques. It also includes information and exhibits that should be in the initial investigation report, with regard to "How the standards manual might be used ?" and "why to do this after reading this section."

1.1.3.2 Feasibility Study

This phase determines the objectives of the feasibility study and who this task belongs to -- analysts or the project team? It lists out the steps required to complete a feasibility study, identifies the scope of the current system, problems and unexploited opportunities in the current system, which may be either manual or automated. It then discusses the major objectives for the new system, and the various methods to gather data and determine how to use the methods. It also helps to estimate the costs of each possible solution, and develops estimates of the benefits and shortcomings of each solution. It presents users and the management views on the above issues and their decision of whether to commit to the analysis part of the project. This phase may be included some related subphases:

- Current physical model: The description of the system as it is now, including the mechanisms used to accomplish tasks (e.g., people, devices).
- Current logical model: The system description in term of functions, processes, and data with the mechanisms removed.
- New Logical Model: The Current Logical Mode with new features added.
- New Physical Mode: The Current Logical Model with the various processes allocated to automation, manual procedures, other mechanisms.

1.1.3.3 General Design

Show the users the view of the application. In this phase, as well as the previous two, users must be directly involved. In this phase, The analysts' imagination, creativity... and initiative are used to their fullest. It also issues the System Flowcharts to develop. In addition, broad specifications that describe how the data is to be processed by the computer will be developed.

1.1.3.4 Detailed Design

The Analysts have to transform from general design specifications to detailed requirements that can be used in implementing the many tasks that make up the system. The final report should include the Procedural Flowchart, record and report layout, and a realistic plan for implementing the design. No major changes should be made until the design is implemented and the system is operational, Then the programs for both transaction processing and batch jobs are executed. If there is the problems, the professional data processing staff is responsible for determining the cause and implementing a solution.

1.1.3.5 Implementation

Detailed logic plans must be developed for all programs before they can be written, tested, and documented. After each procedure and program are tested, the system is tested. The conversion to the new system is made according to a plan developed in the detailed design phase. Many companies encourage their customers documenting their own systems.

1.1.3.6 System Audit

When the implementation report is submitted, an evaluation should be made to determine whether the system meets the objectives stated in the general design report. In this phase, users may be able to suggest the easy-to-implement improvements.

As in the six phase development life cycle, the project can be dropped at any point prior to implementation. A project may be dropped if the benefits derived from the proposed system do not justify commitment of the needed resources. Or if the costs is higher, than expected.

1.2 System approach

Approaches to systems development, in professional organizations, usually follow one of two basic models: the waterfall model or the spiral model.

The Waterfall model is the basis of most of the structured development methods that came into use from the 1970s onwards. It provides a framework for planning top – down systems development. The development flows down a number of successive stages are typically:

- Systems analysis;
- Systems design;
- Systems build and test;
- Systems introduction and transition;
- Maintenance of production status systems.

The Spiral model of systems development proposed by Barry Boehm, has become popular is basis for iterative systems development. The spiral model follows the same breakdown into stages, but takes an incremental approach to systems development. Typically, a system is divided into smaller sub-sets for development and delivery. This provides functionality to end users at regular intervals, rather than at the end of a waterfall development. It is also common in iterative development for highly-skilled developers to model systems with stakeholders, the design and implement them, rather than assign the stages to different departments or groups. Effective use of the spiral approach to system development can deliver systems quickly and ensure user involvement, especially when prototypes are part of the development approach.

1.2.1 System requirements

The system is developed base on the requirements of the system itself (to help manage an organization) and technical requirements.

There are different view points of what information system automatization is like, however, we can classify them into 3 main groups: view point of the system that will be developed, information expert's view point and user's one. These points of view often conflict with one another, at the same time we are required to build up a successful system in which the system, information experts and end users share the same view point. Information system is a system that collects information, manages them and creates information products for its users. The following part will discuss requirements of system, information experts and users.

1.2.1.1 The system's requirements:

- Suitable with the general strategies: Small changes in the organization's development may result in bigger impacts on the information system's requirements. Therefore, during

the system development process, these requirements should be regularly checked for its suitability with the general strategies.

- Supporting decision maker: Together with hands-on experience, knowledge and anticipating ability, information system plays an important role in supporting decision making process;
- Competition edge: The more competitive the environment, the more demand for better systems;
- Return on Investment: A new information system needs to show financial benefits it can bring, because decisions on investment, development costs and operation costs are based on financial analysis; More advanced evaluation techniques can be applied. These techniques take into consideration issues such as customer support, organisational effectiveness, risk, etc.
- Overhead control: human resource change will influence staff number, staff skills and workload. In many cases, while human resource structure is unchanged, workload and requirement for staff skills are yet much higher;
- Supporting operational management: This is clear in preparing detail information, making reports quickly, which contribute to a more flexible and efficient way of management;
- Improving information communication: Optimizing the flow of information, preparing necessary updated information and providing users with the information;
- Impact of information products: Information products are final outcomes of the IT system. We need to pay special attention to requirements for information products so as to thorough analysis. These requirements shall be frequently in comparison with general strategies while developing the system;
- Ability to implement more quickly and better.

1.2.1.2 Users requirements:

Users are those who use the information system to manage their organizations rather than simply those who work with computers. They are the ones who master the current information system (from information sources, management requirements to the system's shortcomings) and are future owners of the system. Thus their requirements should be respected while developing any information system. Attention should be paid in the following issues:

- Easy access: The system must be able to timely access data and manipulation supports.

- The system: The system must be solid and stable, being able to meet staff's requirements and provide accurate information, easy to maintain and restructure, quick in identifying and correcting mistakes;
- Interface: Suitable with working style of users, stable, easy to control data, independent and flexible, enabling users to approach in different ways.

1.2.1.3 Technical requirements:

Technological requirements should be taken into account when designing information systems. The important points are as follows:

- Information volume: Information technology equipment must be suitable with the volume of information that is to be processed;
- Period: Everyday information which arises regularly is repetitive information that requires special care;
- Accuracy: Specially high accuracy is required now and then. Accuracy is important but difficult to meet;
- Complexity: Issues in information treatment can be processed in principle. However, due to its complexity, the current system fails to resolved the issues that need to be resolved by the new system.

Besides the issues of the three view point groups, we would also like to remind you of the following popular issues:

- Incompatibility: Applications developed in different environments are often incompatible. Computers of different kinds are difficult to be connected together, making offices isolated from information processing system;
- Shortcomings: Lack of typical information, unfriendly interface with users, bad storage of information.
- Low reliability: Data is conflicted, inadequate and unamendable, information is not updated regularly;
- Poor resources: Inadequate ability to search for information, lack of exploitation tools for users, low information quality;
- Bad support: Users are not aware of what they have handy, there are no clear development strategies, development pace is slow, support is inadequate, mutual understanding is low.

It is clear that the IT experts and the users view the system from different perspectives thus having different requirements. Analyzer's capability is shown in his/her ability to collect ideas and evaluate them from a wider perspective, because system developers are only knowledgeable of their own areas.

1.2.2 System survey

The survey process is often divided into 2 main phases:

- Preliminary survey: define the project's feasibility;
- Detailed survey: define what needs to be done and accomplishments that should be achieved

Preliminary survey: in this phase, important questions to ask are:

Do we need to carry it out?

What do we need to continue to do?

How long do we need to do it?

What is the estimated price?

What are the benefits and difficulties?

To get the answers for those questions, follow the steps listed below:

- Define what needs to be done to get the same requirements from the organization, users and the information system;
- Define the scale of the problem that needs to be solved in considering particular issues of the organization;
- Define the users whose work will be changed following the development of the system. Users are often divided into 4 categories: users at manipulating level, at supervision level, at management level and professional level;
- Make up a preliminary survey report base on discoveries from initial observation to get an overview from different perspective, making up a solid basis for the next phase.

Detailed survey

The objective of the detailed survey is to find out the best solution in terms of technology, finance, time etc... The outcome of this phase is a report which clarifies the user's requirements, identifies information current, evaluates and picks up the best solution and gives advice users on the current system and how to utilize it in the future. Surveying is divided into 2 parts:

- Detailing the objectives set forth;
- Identifying information sources and information requirements.

To get the best outcome of detailed survey, it's useful to follow the tips below:

- Review all paper and reports of preliminary survey in the previous phase;
- Review all technical terms;
- Make good plan for the survey;
- Give assignment to each group member;
- Supervise closely the implementation of the plan.

It is essential to understand thoroughly the current system to get an idea of how to operate the system, how perfect the system is and examine the cost at running the system. Once you know the current system well, it is easier to document this phase. In fact, getting to know the current system should be deemed a significant responsibility of most of analysts. In order to really understand the system, analysts need to be as capable as a survey expert who is good at application studying. The following information should be made available about the current system:

- Available information source;
- Hardware/software being used;
- Information processing procedure;
- Information interaction (between internal and external information);
- Quantity of information that needs to be processed;
- System's circle and period;
- Information archiving and exploiting tools;
- Report forms being used;
- System staff;
- Costs.

Besides, it is necessary to find out requirements of the unit in which we're surveying and their product requirements

1.2.3 Investigation Methods

Most of the difficulties one can meet in system analysis result from the survey process. Some people perceive incorrectly that the survey process is finished after the questions on the

current system and the future system have been answered and the answers have been analyzed. In fact, all information reflecting the current situation have to be collected, and it requires great effort so as to decide what information to collect and how to collect it. In this section, we'll discuss several popular survey methods.

1.2.3.1 Survey methods:

The contexts in which you make interviews are often different and unpredictable. However, interviews are the main source of information about the future system and the current system. There are 2 main reasons for interview failures: (i) interviewer fails to understand what users say, (ii) bad communications between interviewer and interviewee. The followings are tips for interviewer:

The interview: Before the interview, you should contact the interviewee directly (or through his/her secretary) to set up an appointment and agree with him/her on time, venue and interview objectives. During the interview, if you behave professionally, you will receive the same attitude from your interviewee. Try to attentively listen to the people you're talking to and take notes of all the necessary information you are provided during the interview. The interview paper and minute are always useful for you and your successor because it helps you master the origin of the information you have. It's important to open and close the interview carefully because this may impact the way your questions are answered. When opening the interview, try to do it in a trustful, respective way with your good will. When closing the interview, you should recap the main points of the interview, make arrangement for the following cooperation and leave the issues open for discussion between both sides. Don't make the conversation too lengthy nor prepare too many questions to ask. The best way is you prepare a short list of the main topics you want ask about

Questions: You should carefully choose the kind of question you will ask. As each kind of question serve a particular context, you will suffer if you choose it wrongly. Open questions always create more opportunities for the interviewee to answer, but don't expect too much, not all the answers are good enough. You should take those questions which show your prediction such as "I feel that...", "I sense that..." and use the words that can help emphasize your idea. Learn to know how to be quiet and listen to your interviewers. Don't present a question while (s)he thinks or manage to answer a question, however sometimes it's useful to give him/her a hint. Don't try to ask in a way so as to lead the interviewee to your own direction, you wouldn't get much information that way.

Communications: The language you use and the way you talk about technical stuff should in some way create a close contact between you and the one you're interviewing.

1.2.3.2 Observation methods:

Official observation: It's not a good method to observe every single elements while collecting information to develop the system. The future system you're building up may be deemed to change the current way of working. Moreover, those you're looking at, may feel uncomfortable and may behave unusually, which will affect your survey's quality

Unofficial observation: In order to get an overview of an organization, take a look at its pile of paper and document, interruption of work, unreasonable timing and positive reflection of a good working environment... It's also important to know the quantity and quality of data that need to be processed and predict how they change over the time. Researching through document is the final good method to get important information

1.2.3.3 Questionnaire method:

This method requires your clear instructions to the user. A questionnaire can be designed base on the following points:

- Title: describe the objectives and main contents;
- Data classification: categories of data that will be used;
- Data: contents of the data in each category.

In general, this method is complicated and ineffective for inexperienced analyzers.

It's clear that each method has its own strong and weak points and is suitable for a particular context. However, regardless what method you use, the general principle is: The more information you get about the operation environment of an organization, the more you understand its issues and be able to make realistic questions about the matters you're interested in. Information can be divided into 3 groups: General information of the organization's vertical structure, information about the organization and information about the units that directly relate to the current issues

In this material, an example of library management system of Institute of Information Technology (IOIT) will illustrate through the survey, analysis and design process.

The main purpose of library management is to serve readers. A library management system is required to receive books and magazines number or code them, store and manage them, produce index of document

Main contents of survey of the Library management system are:

- Relations between the current library management system and other library management systems;
- Main jobs of the library management system;

- The jobs that need to be improved by the new computer system.

In this survey, we concentrate on interviewing method.

The interview questions should consist of the followings:

- What are the main jobs of the library management system?
- What is the function of each job?
- How are the jobs currently done?
- Who is responsible for them?
- What are the restraints and difficulties in each job?
- Who does the library serve? and so on

Base on the answer of librarians, we can continue asking them to get a clear vision of the library management system, and all the requirements for the new library management system on computer.

1.2.4 Survey report

Theoretically, survey report should be written in user's language (not necessarily non-technical language). Technical parts for designing should be put into an appendix.

All reports must have a cover page with name of project, its author, address, contact numbers. Cover page is followed by content pages with the following main items:

- Objectives of the system;
- Inter-connection between related departments;
- Details of the current system;
- Future system and sketchy estimate of costs and benefits;
- Advice;
- Time frame and plan for system development;
- General description (non-technical);
- Original document;
- Evaluation of the current system in terms of: organizational structure, technology, information system, users' IT skills, policy renovation...

The following points should be included in the conclusion part of the report:

- Is the document of the current system adequate?

- Have users reviewed and agreed with your point of view?
- Have users been consulted and has analyzer addressed their concern properly?
- Has the whole report been researched thoroughly?
- What functional requirements need further research?
- Have all requirements been reviewed?
- What are the substitute designing solutions?
- What are the project's possible changes?

In the example under question, after surveying the library management system with the interviewing method, we can summarize the process of the library management system as follows:

The library manages two main kinds of documents - books and magazines. Whenever a new document is received, the librarian writes down all information about it into a register of document and lists it into the list of document in library for readers to look up. If it is a new book, the librarian lists on three fields (in a card): book name, book number, and author name. If it is a magazine, the librarian lists on these fields: magazine name, magazine number, and magazine volume. These cards will be stored in library cards, and the readers will look up documents by these cards. There is only one book number, and only one magazine number.

Only staff of the Institute is served by the library. Each staff must come from a department in the Institute, and each staff is provided with his/her own number which never coincides with number of other staffs. Whenever a reader borrows a document, the librarian writes down this information: reader number, reader name, document number, document name, borrowing day and returning deadline into the register of borrowing readers. When a reader returns a document, the librarian looks up for the reader's number, document number in the register of borrowing readers, and deletes it. At the end of each month, each quarter, or each year, the librarians have to make report of the all new documents in the library, the borrowing and returning status, the list of readers who did not meet deadline and did not return the document.

After surveying the library management system with interviewing method, we can conclude that all jobs the system has to do are:

- Receiving new documents, store all information about it;
- Establish document numbers and looking up fields;
- Manage readers;

- Create borrowing and returning ticket;
- Report situation about the documents and readers in the library.

Requirements of the new library management system on computer:

- The form which is used for entering information of documents must be convenient and easy to use. The number of each document (include books and magazines) must be automatically built base on their information;
- The documents can be looked up in many fields such as: document name, author name (for books), volume of document (for magazines), speciality of documents, collection of documents, master keyword, slave keyword, and so on. The speed of looking up documents must be fast and easy to use.
- Manage readers of the library with their departments in such a way that is simple for building reports, easy for managing and searching.
- The new library management system on computer has to make many complicated reports for serving the library management process.
- Readers can look up for documents on their computers which are connected with to the library's computer (through LAN) instead of going to the library and searching for them on the library cards.

1.2.5 Function analysis

The purpose of function analysis is to master users' requirements to the system, i.e. what the system will have to do regardless of how it's going to do it. Thus, the analysis process needs to focus to describe the organization. The description is then sent to users for their comments and approval before your taking of the next steps. A complete function should have the following elements:

- Function name;
- Description of function;
- Function's input (data);
- Function's output (data);
- Events that cause changes and their efficiency;

After the function diagram has been completed (with user's approval), we will have a clearer understanding of the system's requirements though it is not yet a all-sided understanding.

A function diagram just tells us what to do, not how to do. At this point, the diagram has not mentioned issues such as personnel, machinery, equipment and functional requirements as well as the difference between operational function and management function. All functions are equally important and thus being processed as parts of one structure. Functions shown in a diagram in which a upper level general function is divided into functions at lower levels.

Base on the result of the survey phase, we realize that the library management system concerns with other system as the following top – level data flow diagram:



Questions

- 1̃ What is system, give some definitions of system
- 2̃ How do you distinguish natural systems and man-made systems
- 3̃ Please list some automated systems and the rules to build them up
- 4̃ Who participate in system development? Please tell the role of each of them
- 5̃ What is system development life cycle? What are its components? Which part in the life cycle does each of the participating system developers handles?
- 6̃ From what perspectives are the system requirements viewed?
- 7̃ What are the purposes of system survey? Please tell several popular system survey methods
- 8̃ Why is survey report necessary? What is an adequate and detail survey report like?

2

Chapter 2: SYSTEMS ANALYSIS

This chapter deals with techniques applied in information system analysis, data modeling and normalization. This chapter shows a process of providing full specification of systems to users to help them consider and accept. This specification is also a major information source for designers of the new system. It not only specifies the system's objectives but also describes the work and its constraints to which designers have to comply.

2.1 Analysis of structured system

This part discusses the process of system analysis, analysis methods and tools supporting the analysis of data collected in previous surveys

2.1.1 Overview of system analysis

System analysis is a relatively young field in mankind's knowledge but demand for system analysis existed many centuries before the introduction of computers. In mid 19 century, practitioners in labor, organization and methodology had established many improved methods of working. This is the first approach to system analysis.

With the development of information technology, system analysis science also develops more and more vigorously and has a significant role in a life cycle of an IT application and of IT projects in general. At the moment, there is no method that ensures success and that can be viewed as a "right" way for analysis but the application of structured system analysis increases the chance of success for most of typical applications and it proves efficient in a range of analysis in real life.

Until today, system approach is still viewed as a sound foundation for structured system analysis.

Structural system analysis is a modern approach to different analysis and design phrases of the system development process which is accepted because of its strong points over other traditional approaches. The structural system analysis has the following main characteristics:

- The system is developed in the top - down order;
- During system analysis and design, several tools, techniques and models are used to record and analyze the current system and new requirements of users, and define a format for the future system;
- The major tools used in structural system analysis include: function diagram, data flow diagram, data dictionary, process specification, entity relationship diagram;

- Separation between physical model and logical model. A physical model is often used in surveying the current system and designing the new system while a logical model is used in analyzing system's requirements. This is a significant advantage brought about by the structural system analysis method;
- Acknowledging users' role in different steps of system development;
- Different steps in structural analysis and designing can be carried out at the same time rather than in one by one order. Each step can improve the analysis and designing made in a previous step;
- Structural analysis is supported by advanced technology in both hardware and software, therefore system development with this method is less complicated;
- Structural analysis when put together with the prototype method can help users and analysts have an idea of the new system and help make best use of both methods.

Two models of structural system analysis:

Waterfall model: has been a basis for a majority of structural system analysis methods since the 1970s. This model consists of different phases carried out one after the other. Each phase can be assigned to a group of experts

Spiral model: initiated by Barry Boehm, has become more and more popular and a basis to iterative development systems. This approach also consists of various phases carried out one after the other as in the Waterfall model. However, the system development process is divided into different steps and smoothed after repetitive steps, the system becomes more perfect after each repetitive steps. In this model, system developer can hand system's functions over to users more frequently rather than giving them all the functions at the end of the development process.

2.1.2 Nature of analysis

Analysis focus on systems' requirements specification and clarification and is the stage when system designers have to work at two levels of definition regarding the study of situational issues and possible solutions in terms of "what to do" and "how to do".

System analysis process in its most general form includes the following main step:

- Identify the operation of the existing system;
- Understand what the existing system does;
- Understand the needs of the users;
- Decide on what the new system should be doing;

- Decide on how the new system will function.

In the process of system analysis, it is not always possible to study every aspect of each of the above steps and this is the time to pay attention to the following:

- If the method of working applied to participants of system development (users, analysts, designers, and programmers) is efficient;
- If the update of changes that happens in the process of system development is touched upon;
- Tool for analysis;
- If workload allocation is reasonable. (Users have to take part in 2 systems (old and new) at the same time);
- If documents are easy to understand by all participants.

2.1.3 Importance of analysis in system's life cycle

A system's analysis and development life cycle include the following tasks: survey, analysis, design, implementation, system testing and approving, installation and maintenance. Main subjects in the whole life cycle of the system are: users, managers and technical experts including analysis, design and program specialist...)

The primary purpose of the analysis activity is to transform its two major inputs, users policy and system charter, into structured specification. This involves modeling the users' environment with functioning diagrams, data diagrams, entity-relationship diagrams and the other tools of the system analysis.

The quality of system analysis can have a big effect on speed of system designing, the programming and time for testing because 50 per cent of faults in the system originated from shortcomings in system analysis. Programmers can complain about slow speed of work because they have to review analysis and designing. This shows the bad quality of system analysts (because of inadequate experience or improper working attitude). Moreover, speed of system analysis activities is also a very important issues because there are always complaints about time, etc. And the products of the system analysts are often specification description and diagrams of technical nature and these are not highly valued. For users, what they care about is what functions the program can perform if it meets the professional need dictated by the system, if its reliability is prove while testing with real figures, if the interface is users-friendly. Analysis plays a very important part in the life cycle of the system because this activity relates to almost all other activities and all subjects participated in the system.

2.1.4 Role and requirement of system analysts

The system analyst is a key member of any systems development project. In a broader sense, the systems analyst plays several roles:

- Archaeologist and scribe: As a systems analyst, one of the main jobs is to uncover detail and to document business policy that may exist only as “tribal folklore”, passed down from generation to generation of users.
- Innovator: The systems analyst must separate the symptoms of the user’s problem from the true causes. With his or her knowledge of computer technology, the analyst must help the user explore useful, new applications of computers.
- Mediator: The system analyst who often finds himself in the middle of users, managers, programmers, auditors and various other players, all of whom frequently disagree with one another.
- Project leader: Because the systems analyst is usually more experienced than the programmers on the project and since he is assigned to the project before the programmers begin working, there is a natural tendency to assign project management responsibilities to analyst.

This means that, as a systems analyst, you need:

- More than just the ability to draw flowchart and other technical diagrams;
- Skills to interview users, mediate disagreements;
- Application knowledge to understand and appreciate the user’s business;
- Computer skills to understand the potential uses of computer hardware and software in the user’s business;
- Able to view a system from many different perspectives;
- Able to partition it into levels of subsystems;
- Able to think of a system in abstract terms as well as physical terms.

2.1.5 Popular methods of system analysis

In the following presentation, you will be introduced to using a range of tools and techniques that enable analysts to have a better understanding and to find a solution for every single case of application. The usage of tools and techniques forms an approach named structure system analysis. Structure system analysis originated from observations that principles of structure programming can also applied to analysis and designing stage. The moving towards structure system analysis can be explained by the following description:

Graphic: composed of variety of diagrams, supported by detailed textual material that, in many cases, serves as reference material rather than the main body of the specification.

Partitioned: so that individual portions of the specification can be read independently of other piece.

Minimally redundant: so that changes in the user requirements can usually be incooperated in just one part of the specification. This approach is now used in the majority of systems development organizations, as well as a large number of engineering-oriented organizations.

2.1.6 Tools supporting analysis: Functioning diagram, data flow diagram, and relationship diagram...

In the process of system analysis, analysts often construct models to give an overview or stress on aspects of the whole system. This enables the analyst to contact users in the best way and when users' need is changed, it is possible to modify or construct a new model. Analysts use model to:

- Concentrate on important features of the system, pay less attention to less important ones;
- Able to respond to changes or changes in user's requirements with low cost and risk;
- Properly understand users' environment and write documents in the same way that designers and programmers construct the system.

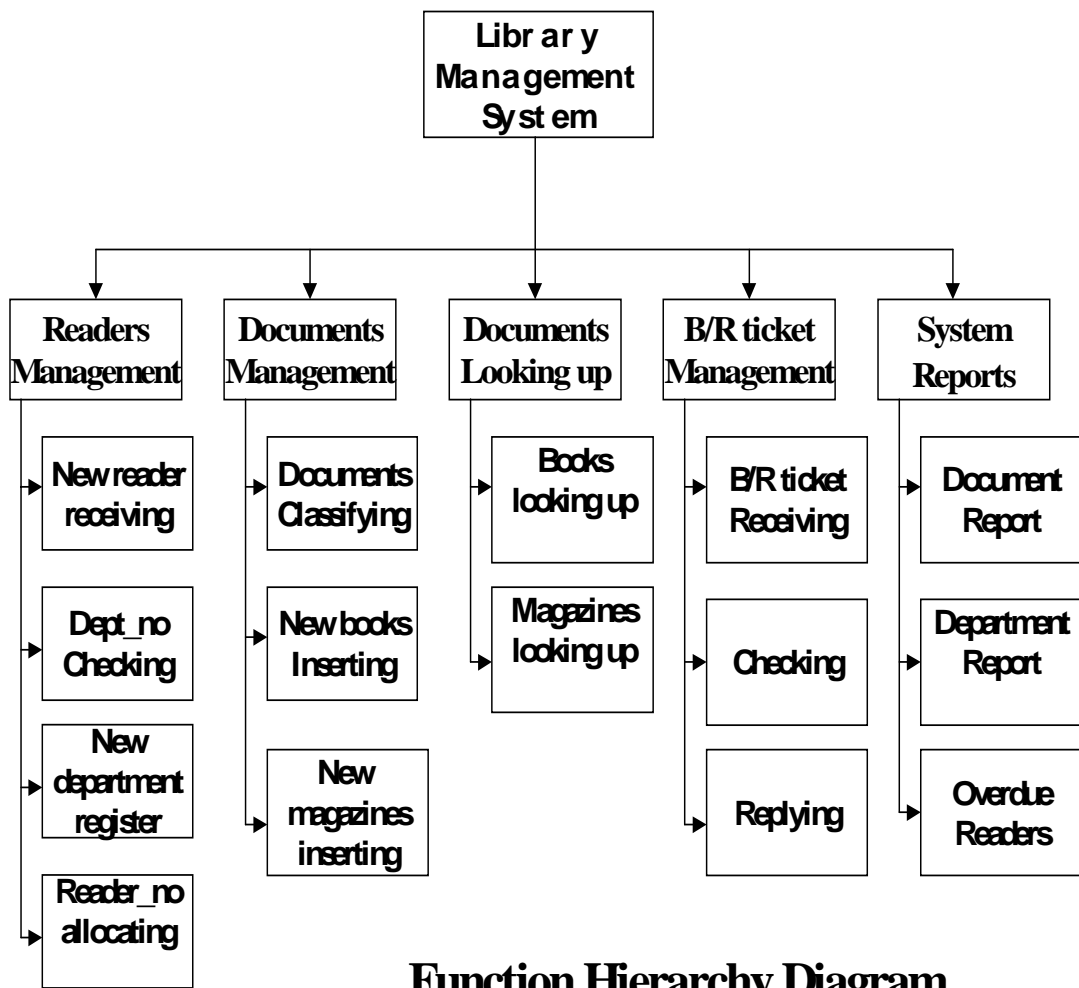
Three important modeling tools used in system analysis is:

2.1.6.1 Functional diagram (FD):

A functional diagram is used to show system's functions that will be constructed and the implementation process of data diagram. Moreover, function diagram will also be used to determine the appearance frequency of smaller process in the data flow chart. If during the construction of functional diagram, analysts identify new functions, the analysts need to determine if it is a wrong move to ignore the discovered functions. It is necessary to decide to add or remove in the most appropriate way. Functional analysis with the help of modeling tools provides important details that will be used often in later stages of analysis. With detailed job description, information processing and exchanging process, input and output of every function will help analysts understand more clearly system's requirements. However, it is necessary to note that function approach to issue is not a comprehensive approach. A function diagram only shows what to do not how to do. In a functional diagram, a function is divided into many smaller functions and each smaller function contains many even smaller ones. Constructing diagram is a process of division, from a higher function to appropriate smaller functions. Diagrams need to be presented clearly, simply, exactly, fully and

balancedly. Function of the same level has the same level of difficulty need to be on the same page.

In the current example, the function hierarchy diagram is as follows:



Function Hierarchy Diagram

2.1.6.2

Data flow diagram: describe the information flow in the system

The next step of system analysis is to consider in detail the information necessary for the implementation of functions discussed above and the one necessary for the improvement of the functions. Modeling tool frequently used for this purpose is data flow diagrams. Data flow diagram will support 4 main activities:

- Analysis: DFD is used to determine requirements of users
- Design: DFD is used to map out a plan and illustrate solutions to analysts and users while designing a new system
- Communication: One of the strength of DFD is its simplicity and ease to understand to analysts and users;
- Documents: DFD is used to provide special description of requirements and system design. DFD provide an overview of key functional components of the system but it does not provide any detail on these components. We have to use other tools like database dictionary, process specification to get an idea of which information will be exchanged and how.

The data dictionary is an organized listing of all the data elements pertinent to the system, with precise, rigorous definitions so that both user and systems analyst will have a common understanding of all inputs, outputs, components of stores, and intermediate calculations. The data dictionary defines the data elements by doing the following:

- Describing the meaning of the flows and stores shown in the data flow diagrams;
- Describing the composition of aggregate packets of data moving along the flow;
- Describing the composition of packets of data in stores;
- Specifying the relevant values and units of elementary chunks of information in the data flows and data stores.
- Describing the details of relationships between stores that are highlighted in an entity-relationship diagram.

The system analysis can ensure that the dictionary is complete, consistent, and non-contradictory. He can examine the dictionary on his own and ask the following questions:

- Has every flow on the data flow diagram been defined in the data dictionary?
- Have all the components of composite data elements been defined?
- Has any data element been defined more than once?
- Has the correct notation been used for all data dictionary definition?

- Are there any data elements in the data dictionary that are not referenced in the functioning diagrams, data flow diagrams, or entity-relationship diagrams

Building a data dictionary is one of the more important aspects and time consuming of systems analysis. But, without a formal dictionary that defines the meaning of all the terms, there can be no hope for precision.

The process specification:

As we know, there is a variety of tools that we can use to produce a process specification: decision tables, structured English, pre/post conditions, flowcharts, and so on. Most of the systems analysts use structured English. But, any method can be used as long as it satisfies two important requirements:

- The process specification must be expressed in a form that can be verified by the user and the systems analysts;
- The process specification must be expressed in a form that can be effectively communicated to the various audiences involved.

The process specification represents the largest amount of detailed work in building a system model. Because of the amount of work involved, you may want to consider the top – down implementation approach: begin the design and implementation phase of your project before all the process specifications have been finished.

The activity of writing process specifications regarded as a check of the data flow diagrams that have already developed. In writing process specifications, you may discover that the process specifications needs additional functions, input data flow or output data flow... Thus, the DFD model may be changed, revisions, and corrections based on the detailed work of writing the process specifications.

Data flow diagram can be described in the following ways:

- What functions should the system perform?
- Interaction between functions?
- What does the system have to transfer?
- What inputs are transferred to what outputs?
- What type of work does the system do?
- Where does the system get information from to work?
- And where does it give work results to?

Regardless of the ways it is described, the data flow diagram needs to meet the following requirements:

- Without explanation in words, the diagram can still tell the system's functions and its information flowing process. Moreover, it must be really simple for users and systems analysts to understand.
- The diagram must be balancedly laid out in one page (for small systems) and in every single page showing system's functions of the same level (for larger systems)
- It is better for the diagram to be laid out with computer supporting tools, because that way the diagram will be consistent and standardized. Also, the adjustment process (when needed) will be done quickly and easily.

The main components of data flow diagram are:

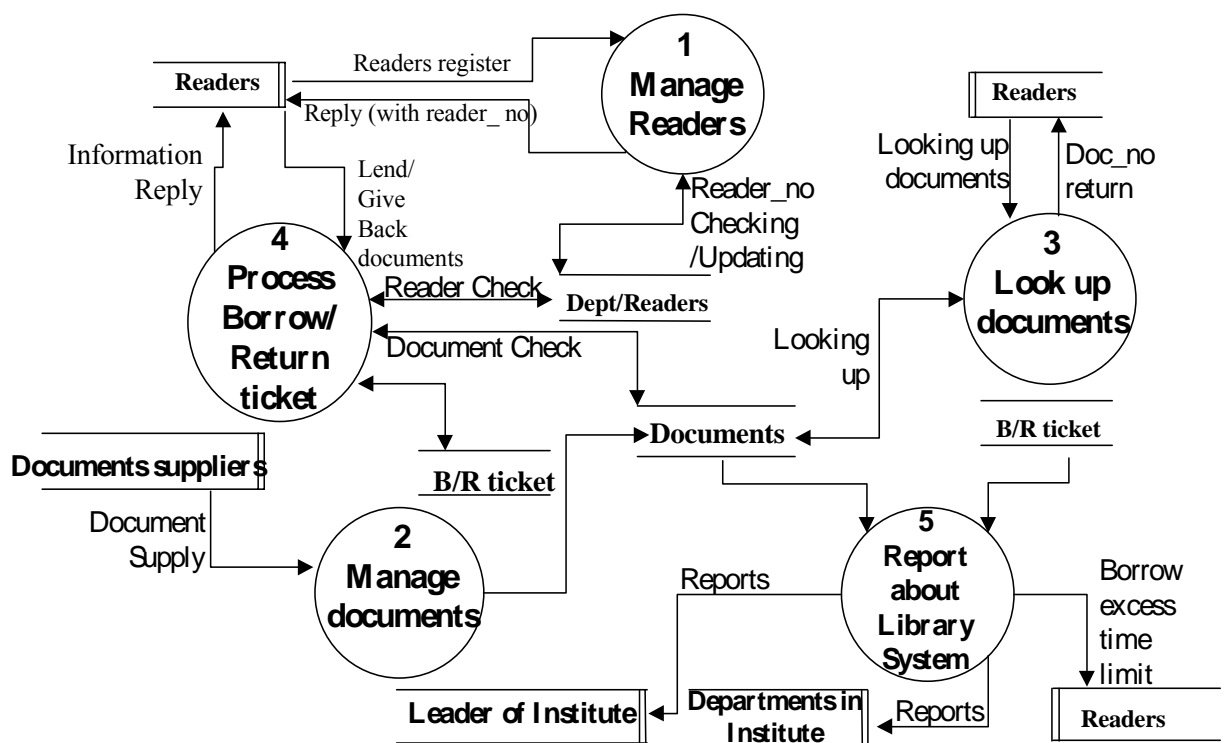
- **The process:** The process shows a part of the system that transforms inputs into outputs; that is, it shows how one or more inputs are changed into outputs. Generally, the process is represented graphically as a circle or rectangle with rounded edges. The process name will describe what the process does.
- **The flow:** The flow is used to describe the movement of information from one part of the system to another. Thus, the flow represents data in motion, whereas the stores represent data at rest. A flow is represented graphically by an arrow into or out of a process.
- **The store:** the store is used to model a collection of data packets at rest. A store is represented graphically by two parallel lines. The name of a store identified the store is the plural of the name of the packets that are carried by flows into and out of the store
- **External factors:** External factors can be a person, a group of persons or an organization that are not under the studying field of the system (they can stay in or out of the organization), but has certain contact with the system. The presence of these factors on the diagram shows the limit of the system and identifies the system relationship to the outside world. External factors are important components crucial to the survival of every system, because they are sources of information for the systems and are where system products are transferred to. An external factor tends to be represented by a rectangle, one shorter edge of which is omitted while the other is drawn by a duplicated line.
- **Internal factors:** While the external factors' names are always nouns showing a department or an organization, internal factors' names are expressed by verbs or modifiers. Internal factors are systems' functions or process. To distinguish itself from external factors, an internal factor is represented by a rectangle, one shorter edge of which is omitted while the other is drawn by a single line.

You can construct DFD model of system with the following guidelines:

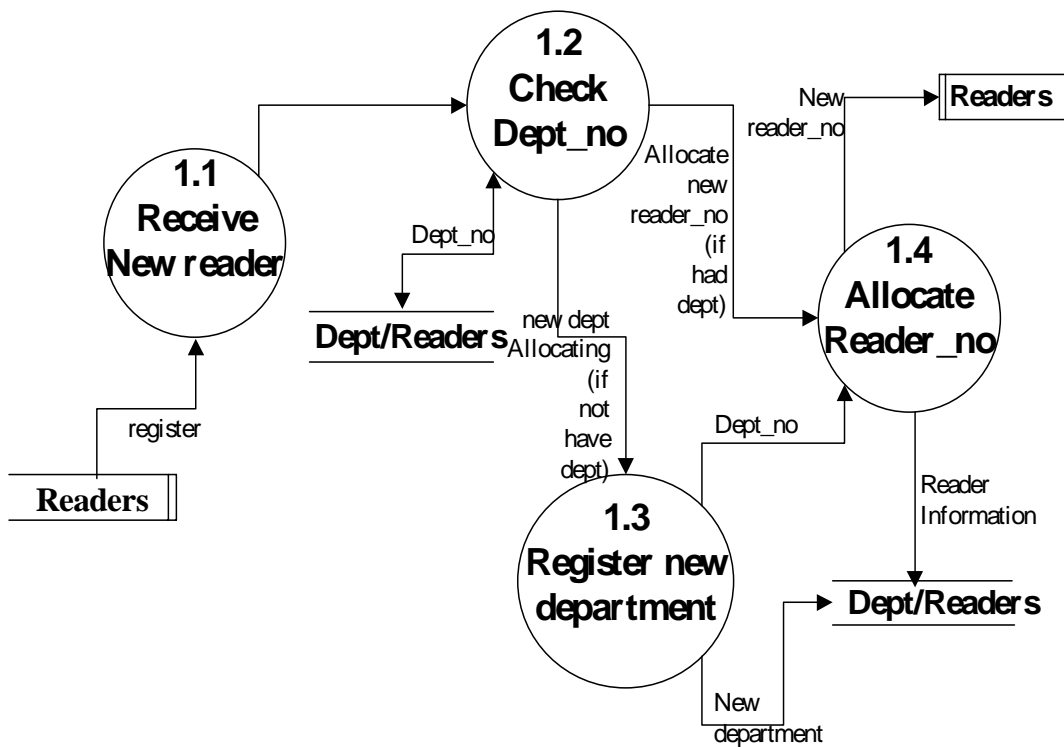
- Choose meaningful names for processes, flows, stores, and terminators
- Number of processes
- Re-draw the DFD many times
- Avoid overly complex DFD
- Make sure the DFD is consistent internally and with any associated DFD

To recap, DFD is one of the most important tools in a structured system analysis. It presents a method of establishing relationship between functions or processes of the system with information it uses. DFD is a key component of the system requirement specification, because it determines what information is needed for the process before it is implemented. Many systems analysts reckon that DFD is all they need to know about structured analysis. On the one hand, this is because DFD is the only thing that a systems analyst remembers after reading a book focussing on DFD or after a course in structured analysis. On the other hand, without the additional modeling tools such as Data Dictionary, Process Specification, DFD not only can't show all the necessary details, but also becomes meaningless and useless.

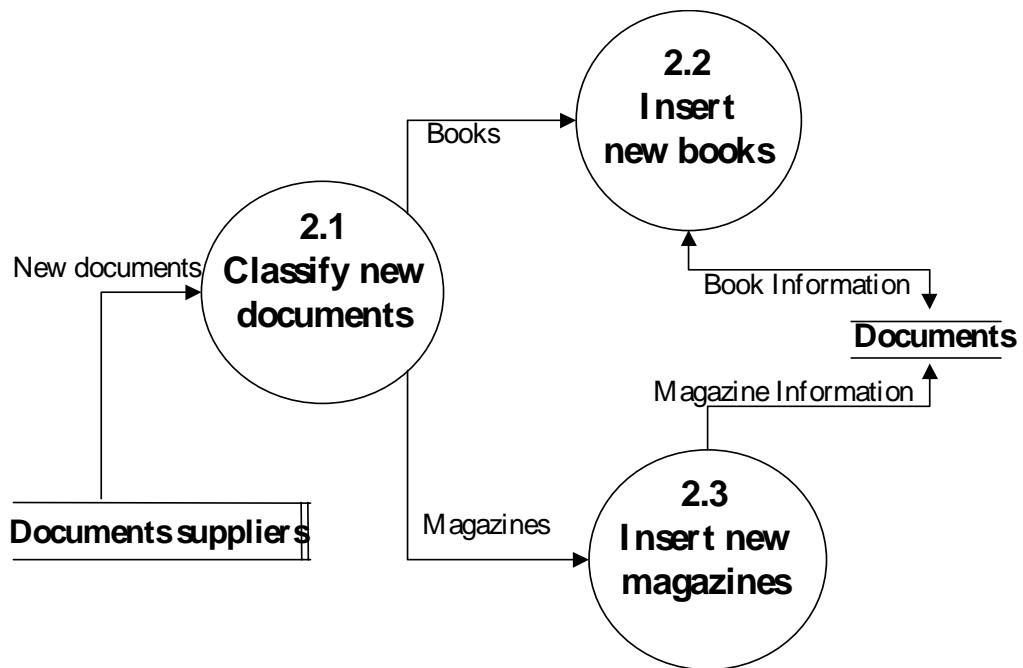
In the example of library management system, corresponding to each level of function hierarchy diagram, we develop the data flow diagrams:



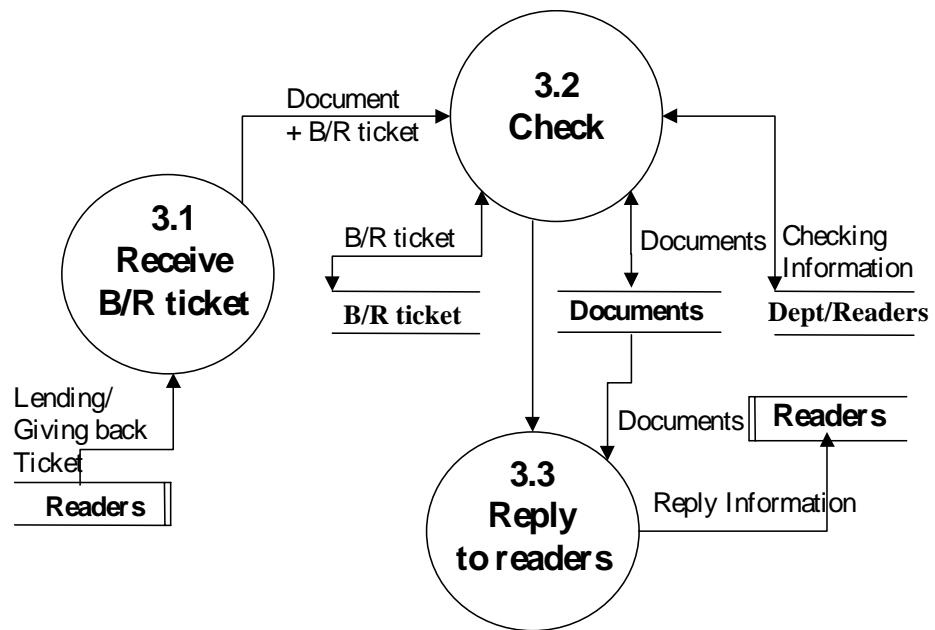
Data flow diagram high level



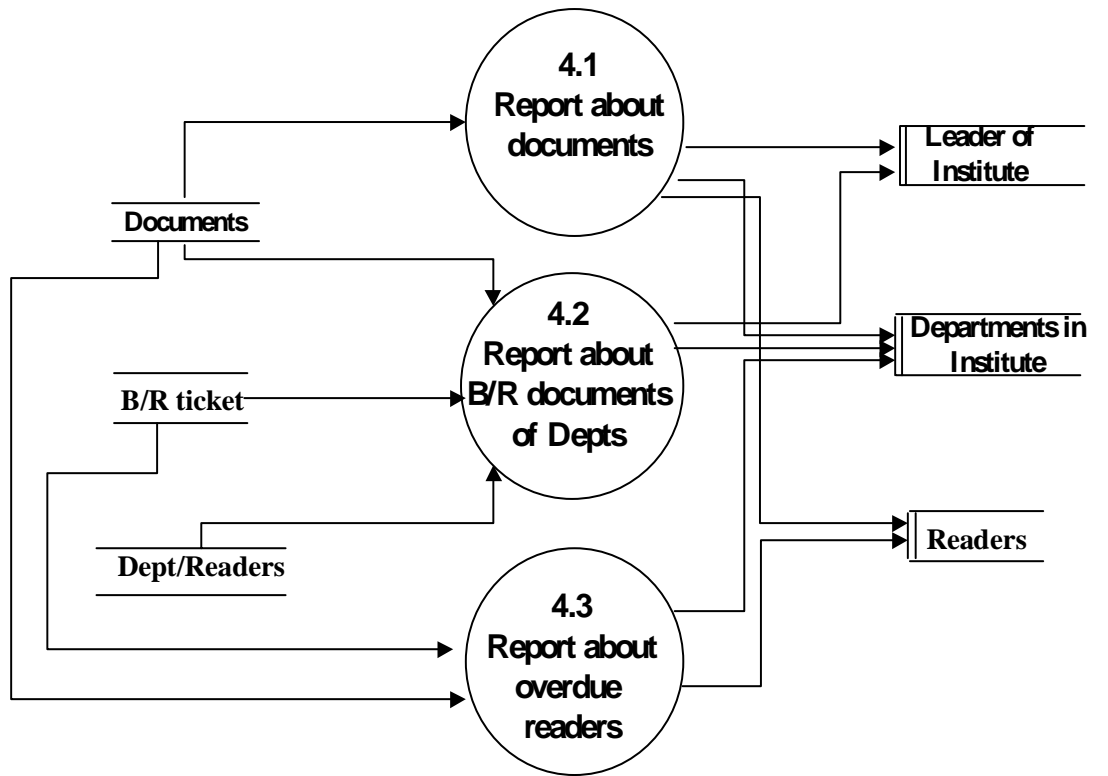
Data flow diagram (Function 1).



Data flow diagram (Function 2).



Data flow diagram (Function 3).



Data flow diagram (Function 4).

Summary:

At the end of function analysis phase, we have 2 diagrams: the function hierarchy diagram and the data flow diagram. They will be a basis for finding out entities and relationships between them (which is defined in the entity relationship diagram).

2.1.6.3 Entity relationship diagram (ERD)

Though the relationship among data store is not emphasized in data flow diagram, it is well reflected in ERD. When developing a new information system, analyst often has discussion with users, especially database manager, on the current system. It's they who have the responsibilities to collect necessary information about the system required by system analyst. ERD is one of the most useful model forming tools to organize this discussion. ERD is network model that describes stored data of a system at a high level of abstraction. For system analyst, ERD has a major benefit: it highlights the relationship between data stores on DFD which would otherwise only be seen in the specification process.

The main components of an ERD include:

- Entity
- Attribute
- Relationship

Entity is a subject, a duty, or an event that has a significant meaning to the future system and is displayed by a rectangle with round corners. Each entity has its own name. In some cases, entities of the same sort can be merged into one entity type. Entity type of a system is defined based on the outcome of system function analysis.

The simplest approach to decide whether a type of information should be put in the model as a type of entity is to make up the following question: Are these information tables useful for the system? If the answer is yes, system analyst has to identify the basic entities that create flows in the data table.

After a suitable type of entity, entity nature have been identified, the next important step is to define the information that needs to be archived for each entity

- Attributes are the characteristics of the entity displayed by fields or columns of a table.

Relationship shows connections among the system's entities. These connections are displayed by triangle headed arrows. There are 3 major types of relationship used in ERDs:

One - one relationship

One - many relationship

Many - many relationship

Let's assume that we have 2 entities in tables A and table B, a one - one relationship will exist between them if:

With each entity in table A, there is a corresponding entity in table B and vice versa, with each entity in table B, there is a corresponding entity in table A. This relationship is displayed by a normal connection line.

Assume that we have 2 entities in table A and table B, a one - many relationships will exist between them if:

With each entity in table A, there are several entities in table B and with each entity in table B, there is one and only one entity in table A. This relationship is displayed by a triangle-headed connection line. The non-triangle headed end shows to the table with one entity and the other end shows the table with various entities

Assume that we have 2 entities in table A and table B, a many - many relationship will exist between them if:

With each entity in table A, there are several entities in table B and with each entity in table B, there are several entities in table A. This relationship is displayed by a connection line with triangle at both ends.

In fact, the many - many relationship is often transferred to the one - many relationship via "connection entity" with 2 upper entities.

Thus, of the 3 types of relationships, the one - many relationship is the most important and popular because the one - one relationship can be integrated in a table, the many - many relationship is transferred to one - many relationship by creating a "connection entity", which facilitates the data modeling

In the example of library management system, we see the following relationships:

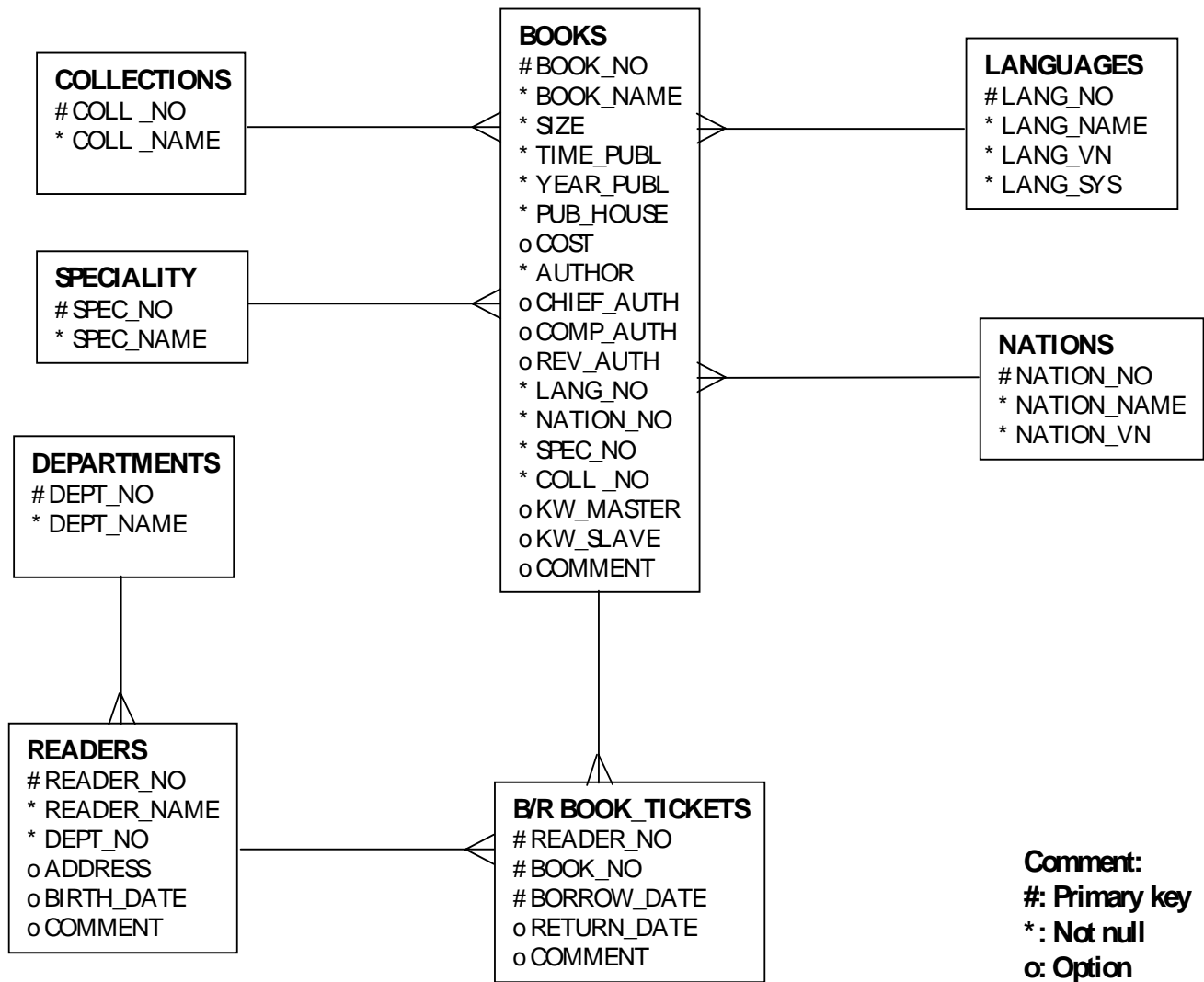
We realize that: one book or magazine (documents) must be belong to one language, but one language can have many books or magazines, so this relationship is one - to - many(one language has many documents). The similar relationships are: documents and nation, documents and collection, documents and speciality.

With the magazines: we realize that each magazine category has many volumes (depend on years, months, and numbers...), but one volume must be belong to one magazine category, so this relationship is one - to - many.

The relationship between department and readers is also one - to - many relationship, because one reader must belong to one and only one department, but one department can have many readers (staffs).

And the last relationship: the relationship between readers and documents: This is a special relationship: one reader can borrow many documents, and one document can be borrowed by many readers at different time (because, when a reader gives back a document, it can be borrowed by another reader again). So we can say that this relationship is “many - to -many” relationship, and separate it into 2 “one - to - many” relationships and one entity, the Borrowing/Returning_Ticket entity, with its primary key is the compose of the primary key of document entity, the primary key of reader entity, and the BORROW_DATE attribute.

So we have the Entity relationship diagrams as follow:



Entity relationship diagram (Books)

Information analysis, data model

At the end of the first part of this chapter, we've discussed in detail information needed to implement the system's functions. The model tools used in the analysis process include Function Diagram, Data Flow Diagram, Entity Relationship Diagram, Data Dictionary, and Process Specification.

This part discusses a totally different approach to information system modelling. This approach is referred to in different names, of which the most popular names are "entity modeling", "data modeling", and logical data analysis". The methodology used in this book concentrates on depicting data analysis techniques and process analysis techniques.

Logical data analysis is an approach integrating the natural data (information) structures used in a system base on an assumption that the system is open to the adequate possibility of information.

Main concepts and methods of data modeling (defining basic information units used in the system and the relationship between them...) are discussed in this part.

2.2.1 Requirements of data approach

Formerly, system analysts who tended to consider system's data from the perspective of current functions, couldn't make use of all the potentials of the information sources inherent in the system. In fact, right after computer system is developed, users would require improvements and amendments to the system, which would then lead to improvements to data structure and organization. It is sometimes impossible to meet these requirements because of analyst's careless approach in the first place. Some suggestions of improvement which sound simple from the perspective of users may be as complicated as to change the structure of a number of data files. In some cases these changes go beyond the physical limit of the initial system solution. Thus, in data analysis, analysts have to pay extra care to users' requirements about the future system.

In short, data analysis is a method to define basic information units useful for the system and identify the internal relationship or the cross identifier between them. This means that any data of the system needs to be identified via supporting tools in previous phases such as DFD, ERD and be archived perfectly, paving the way for the following phases.

2.2.2 Data model and its main components

As mentioned before, the approach being discussed is sometimes called "logical data analysis". In some cases, the word "logical" can be replaced by "ideal". Thus, data analysis means to base on a ideal point of view of data to consider an archive method for it in a perfect

way. However, we also need a solution to depict information that needs to be filed in terms of data structure as well as data organization. Then table was used. Any information in the database can always be archived in simple tables, each table contains several columns and rows. The data analysis technique lies in the way to put which information into which table. Analysts have to be able to point out the type of table needed for the data and the natural relationship among entities in different tables.

Data model displays the expanding of table structure and the relationship between them and the entity relationship model support.

With some tools supporting analysis and design such as Oracle's designer/2000 and developer/2000 and some other tools, analysts can rest assure that (s)he already has good supporting tools for creating and testing diagrams with high reliability.

An entity model (logical data model) consists of the following components:

Entity

Attributes

Relationship

Entity is a thing of significance about which we need to know or hold information

There are 3 types of attributes and any entity can fall in one of these 3 types:

Identifying

Specification

Connection

The identifying attributes or "key" consists of one or more attributes of the entity type used to give a unique identifier to each entity

Specification attribute: Most of the attributes in an entity type can be specification attribute. These are information description to which the entity is identified. These information help system analysts understand more about entity.

It should be noted that each specification attribute occurs uniquely in only one table. Moreover, it's an important duty of analysts to identify and arrange specification attributes in tables perfectly.

Connection attribute or "foreign key" points out the relationship between an existence entity and another entity in another table. The referential integrity rule defines a connection attribute as a column or set of columns that match a identifying attribute in another or the same table.

In this example: As for entity "Book", table "Book" contains attribute "book code" and this attribute occurs again in table "Borrowing list". This shows a relationship between table "Book" and "Borrowing list". It's not necessary to store any other information of "Book" in

table “Borrowing list”. At this point, if users want to get other information of a book such as author, publisher, publishing year, they can search for them in “book code” in table “Borrowing list”.

It is true that data model is used not only as an analysis and designing tool but also a method of checking users’ requirements. If the first direction, from “one” to “many” (one borrowing list can contain many books) is open and un-defined, then the second direction, from “many” to “one” (one book must belong to one borrowing list) is defined, and it shows partly “connection” of requirement specification.

2.2.3 Data model development

The analyst can choose different approaches to develop the initial data model, however, no matter what approach (s)he uses, special attention needs to be paid to describing relationships. The approach discussed in this part is technically the simplest, however how simple it is depends very much on readers’ awareness of entities and the relationship among them. It enables experienced model developers to build up data models quickly, but doesn’t give a clear set of steps that inexperienced developers can follow.

Two steps to develop a data model:

- Defining entity types
- Defining relationships

2.2.3.1 Defining entity types

This step includes defining the major table to store system’s data in. It’s not necessary to define all entity types immediately. In fact, while developing a model, we can discuss and analyze more carefully the entities and smoothen them gradually. The purpose of this phase is to build up initial entity for expansion afterwards. One of the most useful approaches to define initial entities is to use the following 3 type of information to decide which entities need storing details about:

- Information relating to one of the objects of the system (e.g. borrowing lists, returning lists);
- Information relating to the major assets or resources used in the system (such as book store, researching personnel)
- Information about planning and controlling (such as book returning deadline, financial plan)

The second approach is the most popular one: Based on the surveys on the current system and projections of the future system, analysts need to carefully choose the right words to describe the system's operation (its functioning) by making up questions: "Does the system need to store the information about a certain operation? If yes, how does it store the information? Does it just take one row of a table or does it take the whole table?"

Don't worry if some of the important entity types have not yet been defined at the end of this phase, they will be identified in the later phases of modeling.

2.2.3.2 Identifying relationship

After identifying the main entity types, we'll move on to look for the natural connection among them in the one - many relationship.

2.2.4 The advanced data modeling

The definitions of data modeling discussed above will work in most of the common situations of the system once it is computerized. However, there still exist in reality a lot of complicated contexts which require analysts to display them well in models.

Listed below are some contexts where the mentioned modeling techniques are not enough to display in models and yet more techniques are required:

- Optional relationship
- Abstract data type
- Recursive relationships

2.2.4.1 Optional relationship

As discussed in the previous part, the one - many relationship is the most important and popular. Other relationships are either grouped in one table (most of one-one relationship) or made to one - many relationships by creating a connection entity between 2 original entities (most of many - many relationship). However, any popular one - many relationship can be either none or one - many relationship, which analysts should be aware of and be sure to display in model. This means that in most of their life circles, these 2 entities hardly have a connection, the connection attribute will be invalid because during that time, the relationship is not supported by attribute values. Therefore, in the model, this relationship will be shown in a different way depending on which tools used by analysts to support the system analysis process.

2.2.4.2 Abstract entity

While developing models, analysts may find some different entities with similar attributes and share several relationship. In this case, it's better if analysts ignore the differences between the original entities and "abstractize" them to a higher level. In this way, analysts will take an entity prototype that can be used in replace for the similar entities, this will help simplify the model. Although it is necessary in some particular cases to develop data model with the "abstract entity" approach, some of the attributes of the original entity integrated in the new entity type will not be meaningful in all cases (for example, some attributes of "borrowing ticket" entity will be left blank in "returning ticket" entity).

2.2.4.3 Recursive relationship

We've discussed different relationships among entities. For example, the relationship between "borrowing ticket" and "book" are displayed in the data model in the form of one-many relationship and they are stored in different entity tables. However, in reality, there might be relationship between two or more elements of the same entity table. This relationship is called recursive and it is necessary to have a symbol for it.

There are 2 types of recursive relationship: one - many and many - many

a) One - many recursive relationship is a relationship between 2 occurrences of the same entity, but one entity has only one superior entity with which it has a relationship. This type of relationship is normally displayed by the "Pig's ear" symbol. For example a supplier may have a relationship with another supplier where one is a subsidiary of the other. Recursive relationship is supported by the connection attribute.

b) Many - many recursive relationship

This is seldom seen in reality. In this relationship, each entity of a table can have a relationship with some subordinate entities in the same table. And each subordinate entity may also have a relationship with other senior entities. To demonstrate the many-many recursive relationship, we also make it to a form of one - many recursive relationship by using a new type of "connection" entity, similarly to the normal many-many relationship (non-recursive) discussed in previous part.

To recap, in this section, we've discussed the main points of data modeling techniques, implementing methods of data modeling. We've also discussed advanced techniques used in developing complicated models.

2.3 Strengthening of information structure - relationship model

This part discusses detailed analysis of system information to confirm and develop the approach discussed earlier. This part will also discuss techniques including normalization to create a well-structured model.

In this methodology, relationship model is used as a data modeling process in order to test, improve and expand the constructed data model. Relationship model defines a list of all the attributes to every table of entity of the data model. Depending on the supporting tool that analysts use, the attributes form a unique name of the table marked as a key attribute.

The process of creating a relationship model and using it to test data model, includes the following main steps:

- 1 Define the necessary attributes in the to-be-built system;
- 2 Define the type of entity suitable to attributes to limit copying and data redundancy (normalization technique);
- 3 Define the potential relationships within the lists of established attributes for every entity by choosing linking attributes. All linking attributes express the multiple of the one - multiple relationship;
- 4 With known attributes, types of entity and relationship, it is possible to construct a scheme similar to intuition data model. Based on this, a model with the most suitable features is selected;
- 5 After the selection of a data model with the best form to perform system requirements, comes the estimation of the quantity of entity for every table via relationship normalization and this is also done through model.

Despite the possibly time-consuming nature of the implementation of the above steps, efficiency brought over by a careful approach is more important. However, for simple systems, it is not necessary to go through all the listed steps. Experienced analysts need to have their own assessment of the complexity of the system in every specific case.

2.3.1 Identifying attributes

In order to identify all the details of attributes of entities necessary for the system, an analyst can rely on the following resources:

- Interviewing users; (for example, what information the system has to store to control magazines in a library);
- Examining report forms and other documents frequently used in the field under question;

- Based on analysts' experience and knowledge in the field under question, estimation or intuitional identification of attributes is formed.

As thus, for every type of entity in the data model, analysts will have to provide a list of projected attributes. At first, a number of attributes can be seen as belonging to many entities, but they need to be put in suitable lists. This issue will be studied at a later stage of the process with the help of normalization techniques.

In the example of library management system, professional rule analysis gives the following outcomes:

1. The library manages two main kinds of document books and magazines.
2. Each book has a distinct number (for example FV9550). In case there are more than one book with the same book name, they still have distinct book numbers, because they may be received at different times, from different suppliers, and the librarian cannot be sure whether that book was in the library or not.

The way of building distinct number for a book depends on its language (exactly on 3 systems of language: Latin (F), Slav (S), and Vietnamese (VN)); size of that book (divide into 3 sides: Large (L); Medium (V); and Small (N)), and on the number of books on the library. For example: with number FV9550, if the book is written in English (Latin language), the size of the book is Medium (V), and there were 9549 books in the library.

All information about books is stored in a book table which includes a list of attributes:

List of attributes	Type of attributes	Comment
BOOK_NO	Character	Number of the book (must have)
BOOK_NAME	Character	Name of the book (must have)
BOOK_VOL	Number	Volume of book (may have)
SIZE	Character	Size of the book (must have)
TIME_PUBL	Number	Publish time (may have)
YEAR_PUBL	Number	Publish year (may have)
PUB_HOUSE	Character	Publish house (may have)
COST	Number	Cost of the book (may have)
AUTHOR	Character	Author(s) (must have)
CHIEF_AUTH	Character	Chief author (may have)
COMP_AUTH	Character	Compiler author (may have)
REV_AUTH	Character	Revise author (may have)
LANG_NO	Character	Language number(ISO) (must have)
LANG_NAME	Character	Language name(ISO) (must have)
LANG_VN	Character	Language name(in VN) (must have)
LANG_SYS	Character	Language system(ISO) (must have)
NATION_NO	Character	National number(ISO) (must have)
NATION_NAME	Character	National name(ISO) (must have)
NATION_VN	Character	National name(in VN) (must have)
SPEC_NO	Character	Speciality number (must have)
SPEC_NAME	Character	Speciality name (must have)
COLL_NO	Character	Collection number (must have)
COLL_NAME	Character	Collection name (must have)
KW_MASTER	Character	Master Keyword (may have)
KW_SLAVE	Character	Slave Keyword (may have)
COMMENT	Character	Comment (may have)

3. The library lists the magazines regularly by volume, by number or month, by quarter of year, or by year (it depends on publish cycle of each magazines). The library manages 3 kinds of magazines: Information (TIN) magazines, Mathematics (TH) magazines, and technical (KT) magazines. Each magazine category has a distinct number, and each volume of one magazine category also has a distinct number. All information about magazine is stored in a magazine table which includes a list of attributes:

List of attributes	Type of attributes	Comment
MAG_HEAD_NO	Character	Magazine header number (must have)
MAG_DETL_NO	Character	Magazine detail number (must have)
MAG_NAME	Character	Magazine name (must have)
START_YEAR	Number	Having since (must have)
MAG_SHEFL	Character	Where put magazine in Lib(must have)
ISSN_NO	Character	ISSN number (may have)
PUB_HOUSE	Number	Publish house
LANG_NO	Character	Language number(ISO) (must have)
LANG_NAME	Character	Language name(ISO) (must have)
LANG_VN	Character	Language name(in VN) (must have)
LANG_SYS	Character	Language system(ISO) (must have)
NATION_NO	Character	National number(ISO) (must have)
NATION_NAME	Character	National name(ISO) (must have)
NATION_VN	Character	National name(in VN) (must have)
SPEC_NO	Character	Speciality number (must have)
SPEC_NAME	Character	Speciality name (must have)
COLL_NO	Character	Collection number (must have)
COLL_NAME	Character	Collection name (must have)
YEAR	Number	Year of volume (may have)
VOLUME	Number	volume (may have)

NUMBER	Number	number of volume (may have)
MONTH	Number	month of volume (may have)
QUANTITY	Number	quantity of volume (may have)
COMMENT	Character	comment about volume (may have)

4. A new documents will be registered in the finding fields (for readers to look up) in the library card.

5. The library manages readers as follows: In the Institute, there are many departments, sub-departments, and in each department, there are also many staffs who are readers of the library. When a new staff wants to become a reader of the library, a librarian will check if the department (which the staff is working in) is in the list of institute departments or not. If it is not, firstly, the librarian has to register that new department, and then register the number of staff, this number will be the reader_number and will be given back to the staff.

6. When a reader wants to borrow a document, he (or she) has to look up the number of that document on the library cards (depend on the finding fields), then he (or she) has to write the number of document if it is a book, or the number of document and its volume (year, month, number), if it is a magazine, down in a Borrowing ticket. And then he (or she) asks the librarian for borrowing that document. All information in the borrowing ticket is stored in a borrowing/returning ticket table which includes a list of attributes:

If the document is a book:

List of attributes	Type of attributes	Comment
READER_NO	Character	Reader number (must have)
READER_NAME	Character	Reader name (may have)
BOOK_NO	Character	Book number (must have)
BOOK_NAME	Character	Book number (may have)
BORROW_DATE	Date	Borrowing date (must have)
RETURN_DATE	Date	The date that has to return (may have)
COMMENT	Character	Comment

If the document is a magazine:

List of attributes	Type of attributes	Comment
READER_NO	Character	Reader number (must have)
READER_NAME	Character	Reader name (may have)
MAG_NO	Character	Magazine number (must have)
MAG_NAME	Character	Magazine name (may have)
VOLUME	Number	Magazine volume (may have)
NO	Number	Magazine no (may have)
YEAR	Number	Magazine year (may have)
MONTH	Number	Magazine month (may have)
BORROW_DATE	Date	Borrowing date (must have)
RETURN_DATE	Date	The date that has to return (may have)
COMMENT	Character	Comment

With the Borrowing ticket: the librarian has to check whether the document is in the library or not. If it is still in, the librarian will fill information in all fields that has been enumerated in the tables above, otherwise, the librarian has to refuse to lend the requested book.

With the Returning ticket: the librarian only has to receive document and check the reader number, document number in the Borrowing/Returning ticket table and deletes it.

7. At the end of each month, or each year, librarians have to report the document situation in the library, the borrowing, returning documents of all departments in the institute and the list of readers who have been keeping documents overdue.

2.3.2 Determining types of entities (normalization)

Normalization is a process of surveying lists of attributes and applying a range of analysis principles to the lists in order to make them meet the following:

- Minimization of repetition;
- Avoidance of redundancy;
- Elimination of ambiguity.

Repetition is the case when one attribute appears in many entity tables. It only occurs to identifying and connection attributes and is essential in expressing relationships.

Redundancy can be a non-key attribute in many tables or it can point to inferred data. Attributes with values resulted from simple calculation done with other attributes need to be eliminated from the model.

Ambiguity can happen when meanings of attributes are not expressed clearly to groups of users. Normalization process helps analysts study the meaning of every attribute and the relationship model can only be built when a thorough understanding of every attribute is achieved. This process is carried out with the help of “functional dependence” definition (or diagrams). Function Dependence defines a relationship between a pair of fields in each record. In a relation including two attributes A and B, B is said to be functionally dependent on A if, for every valid occurrence, the value of A determines the value of B. In general, with every value of key at any point in time, a table should have only one value. If an attribute is not functionally dependent on key, it should be in another table. From all this, a fully normalized model is the one in which every attribute in the entity table has a Function Dependence relationship to the table’s key attributes.

To carry out normalization, the first task is to select a key for a list of all attributes of an entity. The key includes one or more attributes capable of providing a unique name for every row in a table: no two entities in a type of entity have the same key.

Let’s consider the list of attributes of an entity table put together by an analyst is non-standard (ONF) at the beginning stage. The next level of the list will be the first normal form (1NF).

The first normal form dictates that every attribute must have a single value for any occurrence of its entity at any one point in time. Also, a relation is in 1NF if, and only if, it contains no repeating groups. To be qualified for 2NF which only applies to composite primary keys, every attribute of an entity must depends on the entire identifier of its entity or its value. A relation is in 2NF if, and only if: it is in 1NF and every non-key attribute is dependent on all parts of the primary key. The solutions for both 1NF and 2NF is a non-loss decomposition that involves removing the repeating group and taking its determinant with it.

The nature of the 1NF is to eliminate all repeating groups in a list. By doing this, the key of the table at a higher level will be passed over to the newly created table of the entity because there is one-many relationship between two types of entity and the key is considered as a linking attribute. The new entity then will be studied to find a key to it.

To go from 1NF to 2NF, the analyst must remove any partially dependent attributes from their 1NF entity and create a new entity. After that, copy that part of the identifier of the original entity (on which the removed attributes are dependent) into the new entity. This will probably become part of the unique identifier of the new entity.

The second normal form (2NF): Every attribute must depend on the entire identifier of its entity for its value. 2NF only applies to composite primary keys: a relation is in 2NF if, and only if: it is in 1NF and every non_key attribute is dependent on all parts of the primary key. The solution is again a non-loss decomposition that removing the attribute involved and taking its determinant with it.

To move from 2NF to 3NF, the analyst must remove all independent attributes and put them in an entity of their own. Note that this new entity will now need a unique identifier and is subject to the previous steps.

Third normal form (3NF)

A relation is in 3NF if, and only if: it is in 2NF, and no non_key attribute is functionally dependent on another non-key attribute. This means that, every attribute must not depend on anything except the unique identifier of its entity for a value.

Once again:

- Remove the offending attributes
- Take the determinant along
- The resulting relation is in 3NF
- The initial objective has now been reached, each non-key attribute depends only (and fully) upon the primary key.

3NF is often reached in practice by inspection, in a single step. Its meaning seems intuitively clear; it represents a formalization of designer's common sense. This level of normalization is widely accepted as the initial target for a design which eliminates redundancy.

The characteristic of 3NF is the removal of non-key attribute. This means that attributes dependent on non-key attribute will be removed from the list of entities and make up a new key entity type which is the said removed non-key attribute.

In fact, there are cases where 2 or more 3NF tables of a list of attributes of all entities are put together into one table, the merged table is still 2NF. This is not surprising because the merged table may contain a group of non-key attributes which had never been considered together, and they may contain one non-key dependency. Thus, after grouping the tables together, it's necessary to take 3NF.

In fact, there are some other normal forms that are not widely used. In system analysis, analysts often use a table in 3NF.

To finish the normalization, analyst must double check the following points:

- Is each entity in 0NF, 1NF, 2NF or 3NF?
- Check optionality of relationships.

- Make sure that no two entities have the same unique identifier i.e., they are the same thing.
- Remove “attributes” which are really M to 1 relationships.

In short, to normalize, analysts must finish the following steps:

- Create a list of data items.
- Identify derived items.
- Choose a unique identifier (a “key”).
- Remove repeating groups for that key (and copy across original key).
- Remove “part - key” dependencies (and copy across that part of the original key).
- Remove non-key dependencies.
- Bring together data items with the same key.

When a list of entities does not reach any of the above normal forms, it is necessary to remove one or more attributes inside it, and create more entities to replace. This means that analysts can start with a list of planned attributes for an entity type, after going through the 3 normal forms, (s)he can decide to give some of the attributes in the list to other types of entity. At the end of the normalization, the analyst has not only original entity but also newly defined entity and they all have been fully normalization.

In the example of library management system, the normalization process is as follows:

It is clear that any book or magazine must be published in a language but one language can be used in different documents, thus the relationship between book or magazine with language is a one – many relationship. The same one – many relationship is seen between a document and publishing country...

Firstly, we start with the book table: As we see , the book table has no repeating group and has only one primary key: BOOK_NO, all the other attributes are dependent on this key, so this table has been in the second normal form. The LANG_NAME (language name), the LANG_VN (language name in Vietnamese) and the LANG_SYS (language system) are only dependent on one attribute: the LANG_NO, so we have to normalize them to the third normal form. Similarly with the NATION_NAME , the NATION_VN depends on only one attribute: the NATION_NO; the SPEC_NAME depends on only one attribute: the SPEC_NO; the COLL_NAME depends on only one attribute: the COLL_NO. We have the following table: (the attribute underlined is the primary key).

List of attributes	1NF	2NF	3NF
BOOK_NO	<u>BOOK_NO</u>	<u>BOOK_NO</u>	<u>BOOK_NO</u>
BOOK_NAME	BOOK_NAME	BOOK_NAME	BOOK_NAME
SIZE	SIZE	SIZE	SIZE
TIME_PUBL	TIME_PUBL	TIME_PUBL	TIME_PUBL
YEAR_PUBL	YEAR_PUBL	YEAR_PUBL	YEAR_PUBL
PUB_HOUSE	PUB_HOUSE	PUB_HOUSE	PUB_HOUSE
COST	COST	COST	COST
AUTHOR	AUTHOR	AUTHOR	AUTHOR
CHIEF_AUTH	CHIEF_AUTH	CHIEF_AUTH	CHIEF_AUTH
COMP_AUTH	COMP_AUTH	COMP_AUTH	COMP_AUTH
REV_AUTH	REV_AUTH	REV_AUTH	REV_AUTH
LANG_NO	LANG_NO	LANG_NO	LANG_NO
LANG_NAME	LANG_NAME	LANG_NAME	NATION_NO
LANG_VN	LANG_VN	LANG_VN	SPEC_NO
LANG_SYS	LANG_SYS	LANG_SYS	COLL_NO
NATION_NO	NATION_NO	NATION_NO	KW_MASTER
NATION_NAME	NATION_NAME	NATION_NAME	KW_SLAVE
NATION_VN	NATION_VN	NATION_VN	COMMENT
SPEC_NO	SPEC_NO	SPEC_NO	
SPEC_NAME	SPEC_NAME	SPEC_NAME	<u>LANG_NO</u>
COLL_NO	COLL_NO	COLL_NO	LANG_NAME
COLL_NAME	COLL_NAME	COLL_NAME	LANG_VN
KW_MASTER	KW_MASTER	KW_MASTER	LANG_SYS
KW_SLAVE	KW_SLAVE	KW_SLAVE	
COMMENT	COMMENT	COMMENT	<u>NATION_NO</u>
			NATION_NAME
			NATION_VN

			<u>SPEC_NO</u> SPEC_NAME <u>COLL_NO</u> COLL_NAME
--	--	--	--

Secondly, we normalize the reader table with its attributes and have this table:

List of attributes	1NF	2NF	3NF
READER_NO	<u>READER_NO</u>	<u>READER_NO</u>	<u>READER_NO</u>
READER_NAME	READER_NAME	READER_NAME	READER_NAME
ADDRESS	ADDRESS	ADDRESS	DEPT_NO
BIRTH_DATE	BIRTH_DATE	BIRTH_DATE	ADDRESS
DEPT_NO	DEPT_NO	DEPT_NO	BIRTH_DATE
DEPT_NAME	DEPT_NAME	DEPT_NAME	COMMENT
COMMENT	COMMENT	COMMENT	<u>DEPT_NO</u> DEPT_NAME

Thirdly, we normalize the book ticket_L&GB table with its attributes:

List of attributes	1NF	2NF	3NF
READER_NO	<u>READER_NO</u>	<u>READER_NO</u>	<u>READER_NO</u>
READER_NAME	READER_NAME	<u>BOOK_NO</u>	<u>BOOK_NO</u>
BOOK_NO	<u>BOOK_NO</u>	<u>BORROW_DATE</u>	<u>BORROW_DATE</u>
BOOK_NAME	BOOK_NAME	RETURN_DATE	RETURN_DATE

BORROW_DATE	<u>BORROW_DATE</u>	COMMENT	COMMENT
RETURN_DATE	RETURN_DATE		
COMMENT	COMMENT	<u>READER_NO</u>	<u>READER_NO</u>
		READER_NAME	READER_NAME
		<u>BOOK_NO</u>	<u>BOOK_NO</u>
		BOOK_NAME	BOOK_NAME

And then, the magazine table, we realize that, the list of attributes has a repeating group, it includes these attributes: MAG_HEAD_NO, MAG_NAME, START_YEAR, MAG_SHEFL, ISSN_NO, PUB_HOUSE, LANG_NO, LANG_NAME, LANG_VN, LANG_SYS, NATION_NO, NATION_NAME, NATION_VN, SPEC_NO, SPEC_NAME, COLL_NO, COLL_NAME, and COMMENT, so we decompose it into a new table with a primary key: MAG_HEAD_NO without data loss. The remaining attributes (MAG_HEAD_NO, MAG_DETL_NO, YEAR, VOLUME, NUMBER, MONTH, QUANTITY) are put in another table with a primary key including 2 attributes: MAG_HEAD_NO, MAG_DETL_NO. And we have the below table:

List of attributes	1NF	2NF	3NF
MAG_HEAD_NO	<u>MAG HEAD_NO</u>	<u>MAG HEAD_NO</u>	<u>MAG HEAD_NO</u>
MAG_DETL_NO	MAG_NAME	MAG_NAME	MAG_NAME
MAG_NAME	START_YEAR	START_YEAR	START_YEAR
START_YEAR	MAG_SHEFL	MAG_SHEFL	MAG_SHEFL
MAG_SHEFL	ISSN_NO	ISSN_NO	ISSN_NO
ISSN_NO	PUB_HOUSE	PUB_HOUSE	PUB_HOUSE
PUB_HOUSE	LANG_NO	LANG_NO	LANG_NO
LANG_NO	LANG_NAME	LANG_NAME	NATION_NO
LANG_NAME	LANG_VN	LANG_VN	SPEC_NO
LANG_VN	LANG_SYS	LANG_SYS	COLL_NO
LANG_SYS	NATION_NO	NATION_NO	COMMENT

NATION_NO	NATION_NAME	NATION_NAME	
NATION_NAME	NATION_VN	NATION_VN	<u>LANG_NO</u>
NATION_VN	SPEC_NO	SPEC_NO	LANG_NAME
SPEC_NO	SPEC_NAME	SPEC_NAME	LANG_VN
SPEC_NAME	COLL_NO	COLL_NO	LANG_SYS
COLL_NO	COLL_NAME	COLL_NAME	
COLL_NAME	COMMENT	COMMENT	<u>NATION_NO</u>
YEAR			NATION_NAME
VOLUME	<u>MAG_HEAD_NO</u>	<u>MAG_HEAD_NO</u>	NATION_VN
NUMBER	<u>MAG_DETL_NO</u>	<u>MAG_DETL_NO</u>	
MONTH	YEAR	YEAR	<u>SPEC_NO</u>
QUANTITY	VOLUME	VOLUME	SPEC_NAME
COMMENT	NUMBER	NUMBER	
	MONTH	MONTH	<u>COLL_NO</u>
	QUANTITY	QUANTITY	COLL_NAME
			<u>MAG_HEAD_NO</u>
			<u>MAG_DETL_NO</u>
			YEAR
			VOLUME
			NUMBER
			MONTH
			QUANTITY

And the magazine ticket_L&GB table with its attributes:

List of attributes	1NF	2NF	3NF
--------------------	-----	-----	-----

READER_NO	<u>READER_NO</u>	<u>READER_NO</u>	READER_NO
READER_NAME	READER_NAME	<u>MAG_HEAD_NO</u>	READER_NAME
MAG_HEAD_NO	<u>MAG_HEAD_NO</u>	<u>MAG_DETL_NO</u>	MAG_HEAD_NO
MAG_DETL_NO	<u>MAG_DETL_NO</u>	<u>BORROW_DATE</u>	MAG_DETL_NO
MAG_NAME	MAG_NAME	RETURN_DATE	MAG_NAME
YEAR	YEAR	COMMENT	YEAR
VOLUME	VOLUME		VOLUME
NUMBER	NUMBER	<u>READER_NO</u>	NUMBER
MONTH	MONTH	READER_NAME	MONTH
BORROW_DATE	<u>BORROW_DATE</u>	MAG_NAME	QUANTITY
RETURN_DATE	RETURN_DATE	YEAR	BORROW_DATE
COMMENT	COMMENT	VOLUME	RETURN_DATE
		NUMBER	COMMENT
		MONTH	

2.3.3 Defining relationships

After the entity tables have reached 3NF, the analyst’s challenge is to develop an adequate relationship model which includes the original tables and the new ones created during the standardization process. A useful tool to help the analyst to define all the potential relationships in the relationship model is the entity/key matrix. This is done by surveying each entity type and using its only identifier to search for any “connection attributes” in other entity types. Once a connection is found, it shows the presence of a one - many relationship between the 2 relating entity types. Then a list can be created by all the relationships in the table and a model similar to the data model can be built. The entity/key matrix can be developed as follows: The list of entity types is displayed in columns while the list of key attributes in rows. It’s clear that one key attribute may occur with different entity types, but it can only be displayed in one row. Each matrix case is filled out as follows:

“K” (standing for Key) shows the connection of an entity type and its key attributes. “K” is filled into a cell that is the intersection between a key attribute row and an entity column.

“C” (standing for Connector) shows the presence of a connection attribute in an entity type.

From this matrix, the analyst can be aware of the relationships among the model’s entity types by scanning through the entity types in the column and identify “K” in each column and look over other columns to see whether this key attribute occurs. If yes, the corresponding entity types are marked as the “many” end of the one - many relationship, the “one” end of which is the entity type for observation.

2.3.4 Models comparison

After defining the relationships, entity types and attributes, the analyst must find enough proof points to develop a perfect model. However, there is still a need to compare the 2 models: outline of the intuitive data model and relationship model. Theoretically, they should be similar because they depict the same information system, but in fact they are different. The most note-worthy thing while comparing the models is that the original model covers a larger system. In this case, the analyst needs to base on the survey and analysis reports obtained during analyzing the system to discuss with end users on redefining the system’s scale. Moreover, there occur more relationships between entities. Normally, those relationships newly defined with a new entity during normalization are more important and should be put to the final system. But sometimes they don’t bring about any thing significant and only make the system more complicated. In this case, analysts have to make a wise selection to get the most suitable and effective solution.

To recap: in this part we’ve discussed the process of detail information analysis to expand the vision discussed in the previous part. This analysis approach requires a number of techniques including normalizing to create a tight model. Although these techniques have been discussed, they don’t always meet the requirements of all systems. The more important thing to do is to define a sketchy data model and a relationship model for the future system.

2.4 Completion of information analysis

This part concludes the description of information analysis development. The material on the system analysis process describe in detail tools and techniques used in analysis completion and provide a special description of system requirements that users can evaluate and accept. This special description also serves as the main information source for designers of the system because it explains not only the purposes of the system but also its scope and requirements that designers have to comply with. The main contents consist of:

- Corporation of new requirements
- Supporting tools and materials
- Summary of analysis process

2.4.1 Corporation of new requirements

In this chapter, we have discussed the approaches and supporting tools for the system analysis process, especially the function and data analysis. By developing models, a system analyst defines the system's requirements in most of the cases. However, in some cases, new ideas or requirements in constraint, control, improvements arise from knowledgeable users or the requirements which are found by analysts during the surveying phase. Those requirements should be discussed and agreed in written between users and analyst and should be treated as part of the surveying document and be presented in the system's models. Normally, new requirements are added to the current model from functional diagram and adjusted in other models if necessary. This adjustment must be done carefully to make sure that the model is simply designed. It should be noted that one new requirement can be put into the system in different ways. It is then the analyst's duty to find the simplest and most suitable way from the users' perspective.

In fact, there are cases where the new system is not developed base on any current system because the difference between the old and the new system is so big that there is no need to model the old system. The modeling of the new system is done with the methods discussed in the previous parts of this materials

While analyzing the system, we can see that there are 4 main models making up requirement specification, each model is concerned to one aspect of the system.

- *Functional diagram shows the functions that the system will have to perform, not how to perform.*
- *Data flow diagram also concentrates on functions, but it considers the necessary information to perform the related duties.*
- *Data model and relationship model have hardly anything to do with functions, they concentrate to describe the system's information in different levels and the relationship among them.*

Although each model has its own characteristic and objective, they actually have a very close relationship with each other: These models are the basis for designing other models. Therefore, changes in any model may result in changes or adjustments to secure the conformity among models.

2.4.2 Supporting tools and materials

The major tools supporting the analysis process include:

- Process Description
- Data dictionary
- Seminar (meeting)

2.4.2.1 Process Description

Process description is actually detailize the system's requirements displayed in function diagram and data flow diagram. Some of the tools for process description are: block chart, decision table, structured language... Of those tools, structured language (structured English) is considered one of the most useful and popular tools for process description because it is simple for users and other members taking part in developing the system's life cycle. Structured language consists of:

- Imperative verbs
- Technical terms defined in Data dictionary, relationship model or other models
- Some logical words to make up sentences and structure

The process must be described by using one of the 3 main developed structures in structured programming: Orderly, selective and repetitive

Orderly structure is simply description commands for a process to occur following another process.

Selective structure is used when a decision must be made during the process base on 2 or more selection possibilities. The basic structure of the commands is: IF ... THEN ... ELSE;

CASE...

Repetitive structure is used when there is a need to repeat a series of actions in the process. The basic structure of the commands is: FOR...DO WHILE...

Thus, process description is not only useful in the analysis process but is also used to describe the designing process of the physical system and the programming process afterwards

2.4.2.2 Dictionary

With the development of Information Technology and the generation of professional software, data dictionary is becoming more popular and important to those who is interested in system development. Data dictionaries differ due to the different defining standards, different software and hardware environments, or different users. However, regardless of different methodologies, data dictionaries must present simple explanations, the manner to record process's names, the storage manner and how to search for detail information of the

system. This means a Data Diagram describes the data flows in a Data Flow Diagram, the structure of data packages in flowing motion as well as the data packages in storage. It also means the specification of value and unit of information inside the data flow and data storage, the specification of the information relationships between storage inside the entity relationship diagram

The best way to develop a Data Diagram is to use automatic means (professional programs for data dictionary development) to enter the dictionary's entries, and to check the accuracy and suitability of the data put into the system. In case there is not an automatic means, analysts should use the traditional word processing system to develop a text file for the data dictionary's entries

In short, data dictionary development is very essential and time consuming in system analysis, without it, the analysis outcomes may become meaningless

2.4.2.3 Seminar

The seminar technique is a meeting where analysts present their work in different phases. It gives discussion opportunities to users and the system development team (including analysts, designer and programmer) to improve shortcomings in the system before it is really approved. Presenters must prepare their presentation very carefully and talk about it in a simple, clear and short way. They also have to be aware that this is not a scientific seminar, and that everything they present must be made easy to understand by the users. The key to success of the seminar is the behavior of seminar attendees themselves: any criticism, comments shouldn't be directed to any individual who owns the matter under discussion, they should be directed to the common work. Thus, seminar attendees are not expected to re-design the models (in their own opinions), they are expected to point out the weak points and irrelevant parts and give suggestions to analysts for them to improve those parts. In case there are serious mistakes, analysts should be asked to submit it again after successfully correcting the mistakes. After all models have been tested and approved, system development teams should sign approve and consider the approved models a sound foundation for the following development steps.

In each seminar, attendees have to agree to the following roles:

Coordinator: who is responsible for organizing the seminar; sending seminar invitations and hand-outs to attendees and give them enough time to think of the issues that will be discussed in the seminar. For the best outcome of the seminar, the coordinator has, right in the first place, give clear objectives of the seminar, fix time slot for presenters and direct all the comments to the set objectives.

Secretary: who is responsible for taking notes of all the comments and may be asked to summarize the discussed points. All decisions made during the seminar must be noted down carefully.

Presenters: who are responsible for making presentations as approved by the coordinator. While presenting, make sure to give everyone discussion opportunities. Don't try to protect what you have done. Consider other people including users attending the seminar are those who help analysts make clear of their own requirements. Any interviewers are expected to give constructive comments which can be a hint, a confirmation, an idea from the users' perspective or just a clarification of requirements.

2.4.3 Summary of analysis process

This material has described in detail the tools and techniques that are used to complete system analysis and give a requirement specification to users to review and approve. This specification is also the main source of information for the new system designers. It not only explains the system's objectives but also gives a scale and limitations with which designers have to comply.

Thus, at the end of the system analysis phase, analysts have to give adequate material of system requirement specification which has been approved by users. The material should the name objectives, supporting tools and methods to do the following things:

Contents and scale of system analysis

Summary of management work

Function diagram

Data flow diagram

Data model

Relationship model

Process description

Data dictionary

The result of this phase is the 3 important diagrams: the Function Hierarchy Diagram, the Data Flow Diagram, and the Entity Relationship Diagram. These diagrams are very important, we cannot continue without them during the system building process. They are inputs for the next phase: System Design.

Questions

- 1̃ What is system analysis and structured system analysis?
- 2̃ Please describe several models used to analyse system
- 3̃ Why does system analysis play an important role in the system's development life cycle?
- 4̃ Please explain the roles of system analysts
- 5̃ What are the popular tools used during the system analysis?
- 6̃ What is BFD? Why is it important?
- 7̃ What are the main components of BFD?
- 8̃ What are the main components of DFD?
- 9̃ What is data dictionary? Please tell some of its main components
- 10̃ What is process specification? Please tell some of its main components
- 11̃ Base on which foundation do you build the ERD? What are their main components?
- 12̃ What are some of the popular relationships?
- 13̃ What is a recursive relationship?
14. What are some popular normalizations forms used in building and testing of models?

3 Chapter 3: SYSTEMS DESIGN

This chapter introduces techniques for the design of interfaces, menus, and databases, based on the requirement specification worked out during the analysis phase (functioning diagram, relationship diagram, data flow diagram...). At the end of this phase, you need to identify the borderline between the computer system and human being and find the answer to the question of how to attain the system's objectives.

3.1. The overall designing specifications

By definition, analysis and design are two separate activities. However, in practice, the two development activities are so intertwined that no one can say exactly when analysis ends and design begins. For example, design ideas are often conceived during the preparation of documents such as Functioning diagrams, Data flow diagrams, Entity relationship diagrams, Data dictionaries, Specification process from the formal specification of user requirements.

The design of an appropriate information system requires that analysts understand the goals and objectives of management. They must also be sensitive to changes that may occur to these goals and objectives over time in response to shifts in the competitive environment.

3.1.1 Overview of structured design:

The lofty goals of Structured design are accomplished through the consistent implementation of specific philosophical views. One of these, functionality, is stressed throughout the practice of Structured design. At the heart of this method is the notion that a program, a group of programs, or a group of systems is nothing more than a collection of functions.

Structured design is the result of a desire on the part of its authors to formalize what they consider to be good programming practice. A beneficial side effect for the software engineering is that what is easy to maintain is also relatively easy and inexpensive to construct. Thus, Structured Design is a software design method which closely parallels the current trend in technology toward systems which are once easy to maintain and inexpensive, or at least cost-effective, to construct.

The designer must at first 'see through' the programs, modules, and routines and examine relationships. One must temporarily forget about how the systems at hand might be implemented and treat it as a collection of abstractions-logical functions. This gives the software designer a maximum freedom in examining relatively system architectures. The mapping logical functions in to physical modules are delayed until the late stages of design effort.

The outstanding unique aspect of Structured Design is that it includes several different ways of evaluating a design. (Another is that only one the design quality criteria into it may be applied to any software design, not just those produced using this method.

Structured Design's evaluation criteria challenge software engineers in three ways:

- The suppression of coding issues, making these issues subservient to design principles.
- Acceptance of system view and not the local view as key indicator of success or failure
- Resolution of issues related to 'finishing' the design though application of predetermined value systems.

Structured design provides two effective means of evaluating software design decisions. One measures the relative effect of the decision on module quality, while the other measures the relative effect of the decision on the relationship between modules.

The introduction of Structured Design can only work when it provides all concerned with a common value system, that is, a uniform view of what is or is not desirable in a system. Although the approach may differs from organization to organization, followings are some guidelines to maximize the benefit of introducing Structured Design into an organization:

- Make it clear from the start that Structured Design is not a remedy for any problems of the organization
- Obtain support from anyone who may be affected by this change
- Emphasize the flexibility that Structured Design provides and downplay the misconception that it is restrictive
- Structured Design development passes through two primary phases: Logical Design development and Physical Design development.
- Logical Design transforms the dataflow diagrams developed during the Structured Analysis phase into refining the design to reflect the kinds of quality attributes that Structured Design encourages.

- Physical Design phase involves the modification of the Logical design to accommodate whatever changes necessary

As a process, design goes through the following stages:

- Transformation of the problem model into an initial solution model
- Embellishment of initial design to include necessary features
- Transformation of the structured English into pseudocode
- Evaluation and refinement of the qualities of the design
- Revise the design to accommodate implementation constraints

Design is an exact blueprint of what will be built and a basis for the configuration and content of that blueprint.

3.1.2. Design goals and objectives:

The design objectives specified in the user implementation model are the quality of the design. The ability of the programmers to implement a high-quality, error-free system depends very much on the nature of the design created by the designer; also, the ability of the maintenance programmers to make changes to the system after it has been put into operation and to the expanded system depends on the quality of the design.

The field of structured design contains a number of guidelines that help designer determine which modules, and which interconnections between the modules will best implement the requirements specified by the systems analysis. The most important guidelines are coupling and cohesion:

Cohesion: is the degree to which the components of a module are necessary and sufficient to carry out one, single, well defined function. In practice, this means that the systems designer must ensure that they does not split essential processes into fragmented modules and the systems designer must ensure that they does not gather together unrelated processes (represented as processes on the DFD) into meaningless modules. The best modules are those that are functionally cohesive. The worst modules are those that are coincidentally cohesive.

Coupling: is the degree to which modules are interconnected with or related to one another. The stronger the coupling between modules in a system, the more difficult it is to implement and maintain the system, because a modification to one module will then necessitate careful study, as well as possible changes and modifications, to one or more other modules. In practice, this means that each module should have simple, clean

interface with other modules, and that the minimum number of data elements should be shared between modules.

As we have seen, the first step of designing a process is to map the essential model of user requirements onto a configuration of processors. Then, within each processor, the designer must decide how to allocate processes and data to different tasks. Finally, we must organize the processes within each task into a hierarchy of modules, using modeling tool.

3.1.3. Relationship between system analysis and design

A big advantage of using Structured Analysis as a precursor to Structured Design is that nothing developed during the analysis is wasted. Each of the major deliverables from analysis (i.e., dataflow diagrams, data dictionary, structured English, and entity-relationship diagrams) is used in the Structured Design phase to develop one of its major deliverables (i.e., structure charts, design data dictionary, pseudocode, and database design)

A poor analysis job is going to cause problems, whether it was done by the same person doing the design or not. Some of the most common failings during Structured analysis are shown in figure together with the symptoms they produce during design

Problem	Effect on Design
Dataflow diagram not balanced	Interface Problems
'Fuzzy' process name	Need to repartitioning
No dictionary	Duplicates and aliasing
Dictionary not correlated with dataflow diagrams	Duplicates and aliasing and interface problems
Dataflow diagrams not correlated with dictionary	Duplicates and aliasing and interface problems
Pseudocode not validated	Logic errors
Pseudocode not correlated with dictionary	Data aliases, interface problems

	development phase		
Type of model	Structured Analysis	Structured Design	Implementation
Process	DataFlow Diagrams Dictionary Pseudocode	Structure Chart Dictionary Pseudocode	Code Dictionary
Information	Entity-Relationship Diagram Dictionary	Database Design Dictionary	Implemented Database Dictionary
Event	Event Model Dictionary	Event Model Dictionary	Queuing Model Dictionary

3.1.4. Principles of procedure design

A well-designed procedure is clear, unambiguous, complete and concise. This means that vocabulary to describe how the task is done is kept simple and terms chosen have only one interpretation. Procedures should cover every eventuality, such as errors that might occur and how to correct them, or directions for shortening procedures or skipping procedural steps when processing exceptional loads or under emergency conditions. All steps of a given procedure should assume the same base level of knowledge or experience on the part of user. When procedures require actions by two or more persons, the boundary and domain should be well defined.

When designing procedures, analysts should ask themselves if a given task is one that requires human processing. People dislike repetitive monotonous tasks – the very type of work at which computers and peripherals excel. Where humans have a comparative advantage over computer processing is in areas that require adaptability, the ability to make judgements, to handle unexpected events, to make creative decisions. A well – designed system is one in which the tasking provides for optimal division of labor between human and computer. Procedures for humans should take advantage of human skills, allow flexibility of action, and help employees do their jobs so that they are motivated to follow them. During design process, exception of demeaning or inconvenient procedures

is very importance. In practical, exist some following other guidelines for procedure design:

- Standardize procedures whenever possible.
- Be sure new procedures are compatible with existing procedures, not conflicting or counterproductive.
- Guard against procedures that will evoke resistance or negative reactions from employees.
- Avoid assigning a single individual responsibility for two procedures that similar, yet different.
- Allow for feedback and evaluation.

3.1.5. The stages of design

The activity of design involves developing a series of models, in much the same way that the systems analyst develops models during the systems analysis phase. The most important models for the designer are the systems implementations model and the program implementation model. The systems implementations model divide into a processor model and a task model

The processor model: At this level, the systems designer decides how the essential model should be allocated to different processors and how those processors should communicate with one to another.

As processes must be assigned to appropriate hardware components, data stores must be similarly allocated. Therefore, the designer must decide whether a store will be implemented as a database on this processor or another. Since most stores are shared by many processes, the designer may also have to decide whether duplicate copies of the store need to be assigned to different processors. An important point for designer: processor to processor communication is generally very much slower than communication between processes within the same processor. Thus, the system designer will generally try to group processes and stores that have a high volume of communication within the same processor.

During designing a process, the systems designer should raise the following major issues:

Cost: Depending on the nature of the system, the systems designer must choose the most economical solution for each practical case: a single processor implementation or group of processors.

Efficiency: The systems designer is generally concerned with the response time for computer systems. Therefore, the designer must choose processor and data storage devices that are fast enough and powerful enough to meet the performance requirements specified in the user implementation model.

Security: The end user may have security requirements that dictate the placement of some processor and sensitive data in protected location.

Reliability: The end user will normally specify reliability requirements for a new system; these requirements may be expressed in terms of mean time between failure or mean time to repair, or system availability. Alternatively, the designer may decide to implement redundant copies of processes and/or data on multiple processors, perhaps even with spare processors that can take over in the event of a failure.

Political and operational constraints: The hardware configuration may also be influenced by political constraints imposed directly by the end user, by other levels of management in the organization and operating all computer system.

The task model

When the processes and stores have been allocated to processors, the systems designer assign processes and data stores to individual propose tasks.

The program implementation model

At the level of an individual task, the systems designer has already accomplished two level of process and data storage allocation. Within an individual task, the computer operates in asynchronous fashion: only one activity at a time can take place. The most common model for organizing the activity within a single, synchronous unit is the structure chart, which shows the hierarchical organization of modules within one task.

3.1.6 Tools and Techniques of design

The design of information systems requires that systems analysts understand how information flow in an organization, how it relates decision making, and how it contributes to organizational goals and objectives. That is why systems analysis and systems design is inextricably linked. While data are being collected on the problem environment and the analyst is learning about the information needs of the user, rough ideas about the way to improve the information function are being formulated in the analyst's mind. The analyst may begin to sketch these ideas as flow charts or data flow diagrams. However, not until all data are collected and information problems are thoroughly analyzed can preliminary ideas about the design of the new system be refined. The analyst can then focus on design

specifies such as input/output subsystems, processing, and data base design. Most analyst continue to use structured techniques to help them plan system components and their structure, choosing appropriate tools and techniques to aid them in the design process. The completed design should lead efficiency in satisfying users needs and also facilitate maintenance during the life cycle of the system. Each design tool has good features and bad. Selection of design methodology will depend on the background and experience of analysts (designer), the size of the development time, resources allocated to development, the time allowance for the project, and the type of application under development. When choosing a specific tool for a specific task at hand, the following main questions should be considered:

- Will it help the team arrive at an understanding of the system under development;
- Is it easy to learn? Easy to use?
- Will it serve user-analyst communication?
- What does it cost?
- What data structures does it use?
- What data flow and control features does it have?
- Is the technique manageable?

There are many ways to approach system design and many tools and techniques that contribute to the design process. In this part we will find a discussion of still other methodologies that are comely used by analysts. Many of these are complex, so what appears here is merely a brief introduction to their features, benefits and limitations. Unfortunately, the scope here limits the number of design methodologies that can be presented. In the system development process, many design methodologies already exist, but one of the challenges of systems design is to select tools and techniques that are appropriate for an application under development from the wide range of available options. The main selection that follow: ISDOS, Pseudocode, Structured design, the Jackson Design methodology, HIPO, Warnier – or methodology, SADT, Walkthroughs, Entity Relationship model, Data Structure Diagrams, Sematic Data Model and CASE*Method (Oracle).

1. **Information System Design and Optimization System (ISDOS)** will be able to generate system specifications from user requirements recorded in a machine-readable form, design an optimal system to meet these specifications, and construct code for operational system.

ISDOS begins its role in systems development once the information problem has been defined by the user and analyst and stated in PSL (Problem Statement Language), a structured format. It allows the people who define the problem (user or analyst) to state what is wanted without having to specify how these needs should be met. It is not a programming language but one that lets the user state what outputs the system should produce, what data elements the output should contain, what formula should be used to compute output values, what input the system will use, and so on. It is a language to describe systems. A computer program called PSA (problem statement analyzer) then analyzes input in PSL to ensure a complete and error-free statement of the problem. A coded statement of the problem is then sent to SODA (System Optimization and Design Algorithm), software for the physical systems design phase, which generates specifications for the actual construction of programs, for hardware, for the database, and for storage structures.

The two ISDOS modules to receive this output are the Data Re-organizer who constructs files, storing data on selected device in the form specified, and the Code Generator, which organizes the problem statements into programs. Finally, a module called the System Director produces the target information processing system, using the generated code, the stored data and the timing specifications as determined by the physical design algorithm.

ISDOS has been under development for more than a decade. Whether it will ever succeed in optimizing systems design remain uncertain. But progress is being made towards the automation of some development stages.

2. Pseudocode:

Pseudocode can be used to describe an algorithm. Although pseudocode resembles structured English (SE) in using a restricted subset of English, it may be coded and more closely resemble a programming language. Whereas SE is a useful tool when analysts communicate with end – users, pseudocode is oriented towards analyst – programmer communication, describing program logic, using program construction. In practice, there is no standard, universal pseudocode, but all pseudocode has three basic structures: Sequence, Selection and Iteration.

3. Structured Design (SD)

Nature of SD is achieved (implemented) by dividing the system in independent modules (separate pieces) that can be designed, implemented and modified with no (or little) effect on other modules of the system. Coarse (tho) program structure, based on DFD, is

depicted by means of a structure chart. This structure chart, which resembles an organization chart, show relationships between units or modules, and how modules are combined to achieve systems (organization) and design goals. This coarse program structure is then factored, or decomposed, into a fine structure which forms the basic for implementation. In order to choose among alternatives when dividing systems into modules, it useful to evaluate the connection between them. If there are few or no connections between modules, then it is easier to understand one module without reference to others. The notion of module independence can be described in terms of ‘coupling’ and ‘cohesiveness’. These concepts were introduced by Edward Yourdon and Larry Constantine who are concerned with ‘goodness’ of design.

Coupling is the strength of relationships between modules (the degree to which modules are interconnected with or related to one another). The stronger the coupling between modules in a system, the more difficult it is to implement and maintain the system, because a modification to one module will then necessitate careful study, as well as possible changes and modifications, to one or more other modules. In practice, this means that each module should have simple, clean interface with other modules, and that the minimum number of data elements should be shared between modules.

Cohesion is the measure of the strength among the elements in the same module (the degree to which the components of a module are necessary and sufficient to carry out one, single, well defined function). In practice, this means that the systems designer must ensure that they does not split essential processes into fragmented modules and the systems designer must ensure that they does not gather together unrelated processes (represented as processes on the DFD) into meaningless modules. The best modules are those that are functionally cohesive. The worst modules are those that are coincidentally cohesive. High cohesiveness occurs when all of the module parts contribute directly to the purpose or function which the module is supposed to accomplish.

4. Jackson Design Methodology (JDM)

The JDM is a three-step design technique. In essence, it decomposes the design process itself.

- First, the problem logic is structured. Since it is Jackson’s premise that a good reliable program is closely dependent on its data structure, the analyst first defines data structures. That is, the data components are identified, the relationships between these data components are defined, and then one-to-one correspondences between data structures are defined.

- Next, the analyst structures program logic based on the data structures. To do so, the analyst defines the necessary processes and relationships between these processes.
- Finally, the analyst defines the tasks to be performed in terms of elementary operations and assigns these operations to suitable components of the program structure.

This methodology is not a top – down method for developing computer systems. The structure charts are drawn only after action at the lowest level has been defined. The strength of the JDM is that it results in an exact reflection of data structure in the program structure. The adaptations Jackson has made to basic structure charts are simple, but yield (effectiveness) a greatly expanded expressive capacity. Also, the extension is compatible with the notion of structure programming. However, the method is not convenient for large complex systems since it may prove difficult to derive a reasonable program structure from all data structures. A higher level structuring technique in such situation may be necessary. Since Jackson system development is a very detailed and comprehensive method, well – trained analysts are required to use it. When there are pressures to develop a new system quickly, the considerable time and effort required by the method are considered disadvantages. A main criticism of JDM is that it assumes that all important structures of the problem can be seen as sequential processes.

1. Hierarchy Plus Input, Process, and Output (HIPO)

HIPO is a graphic technique that can be used to describe a system. A series of drawings are prepared by analysts that show the function of the system starting with general overview diagrams, then proceeding to detailed diagrams of each specific function. Originally developed by IBM as a technique to document functions of programs, HIPO is today commonly used as a design tool during systems development. The package of HIPO diagrams begins with a hierarchy diagram, called a Visual Table Contents. This diagram, which is similar in tree-like structure to an organization chart, defines the basic functions to be performed by the system and decomposes those functions into subfunctions.

Preparation of HIPO diagrams begins with a series of meetings with the user about general expectations for the system and desired output, individual functions – circumstances under which the function will be performed, data that will be acted upon, processed and expected output, relationships between those function. After that, HIPO diagrams are generally built from the middle up and down. Once the hierarchy diagram has been drawn, analysts review the system from the top down, looking once again at the different functional levels to see if true functional decomposition exists and whether revisions to the structure of the

hierarchy are necessary. A major advantage of HIPO diagrams is their simplicity, their effectiveness as a communication tool, and the ease with which analysts can learn how to diagram systems using HIPO notation. The diagrams provide a structure by which the system can be understood and a visual description of input, functions to be accomplished by a program and output. The primary disadvantage is that it is impractical to draw HIPO diagrams for very large systems.

2. Structured Analysis and Design Technique (SADT)

SADT is a technique to develop large and complex systems. This technique helps analysts think in a structured way about system activities, data and their interrelationships. A precise notation is used to communicate analysis and design results. The method also provides for the division and coordination of effort among the development team and for planning, managing and assessing progress of the team effort. SADT analyzes a system under development from the top down, decomposing it systematically into subsystems, creating a hierarchical parent-child structure. This top – down approach resembles the decomposition of data flow diagrams. SADT is used for both systems analysis and design. Its strength is not so much the separate representation of data and activities, but the ability to diagram the relationship between the two. With SADT the system analyst and designer, management and users can review the project top – down, or focus on a single level of detail. The technique is frequently used in large and complex development project. The major disadvantage of SADT appears to be its richness. The amount of information that SADT diagrams contain may overwhelm the user and prevent their understanding of the system as whole.

3. Entity Relationship Model (E-R Model)

E-R Model is used in data base design helps describe how entities in an enterprise are related to one another. The model recognizes that two sets of data may have a ‘one-to-one’, ‘one-to-many’, or ‘many-to-many’ relationships. After the model is drawn, the analyst looks for semantic synonyms (one entity referred to by two different names) or semantic homonyms (two different entities referred to by the same name) that need correction. Perhaps an entity should be named by a more generalized term and some entities can be combined into a new single entity. In making improvements to the model, the analyst is fine-tuning the logical schema of the data which will ultimately contribute to greater processing efficiency.

4. Data Structure Diagram (DSD)

The DSD is another diagram of pictorially representing data relationship that is used in data base design. The DSD notation expresses relationships among records-the content of the record is documented in a data dictionary. Record names appear in boxes connected by arrows to show relationships – a one – headed arrow represents a ‘one’ relationship, a double – headed arrow represents a ‘many’ relationship.

5. Semantic Data Model (SDM)

The SDM is another useful model for logical data base design and documentation. The model might be compared to pseudocode, but instead of expressing the structure of programs, it enables an analyst to describe the structure of data. For using SDM, the analysts must learn the terminology of the model, its rules and default values. The SDM has the following main advantages:

- The model provides a vocabulary for expressing the meaning of the data. An informal description of each record is allowed and each data item within each record is defined.
- Each attribute is precisely defined, which is essential information for understanding the data and for later programming.
- SDM allows relative data definitions.
- Allowable ranges of values of data items can be identified in the model

1. CASE*Method

This is the structured development technique used by ORACLE consultants, and design is one of its stages. It therefore specifies the input to our design, provides a framework for its execution and defines our target output. It comprises a number of distinct stages, each having a set of tasks and deliverables. Once checked for quality and consistency, the output of one stage becomes the input to the next. Errors are thus trapped as early as possible. Throughout the development, the information gathered is expressed in the form of models. The stages of CASE*Method are:

- Strategy (establish direction for information system development): A strategy study is performed in order to develop an understanding of the business and to establish the scope of the system, its priorities and constraints. The resultant business model describes the information and the functions to be performed.

- Analysis (establish the detailed components of the targeted business areas): Detailed work on the business model provides complete description of the required system.
- Design (establish a detailed technical solution): A model for a specific implementation is developed, based on the business model. It consists of two parts: a database design to fit the information model, and application designs to fit the function model.
- Build and User Documentation (create a functionally working solution and explain the solution in the user environment): The specifications produced are implemented. Documentation is developed alongside the application system.
- Transition (implement the solution in the user environment): The transition to the new system is based on a plan which evolves through the preceding stages.
- Production (keeps the solution working): The system goes live.

Design was conceived in the mind of the analyst but sketched helped document ideas and communicate them to members of the development team and end-users. Design is mental process, a kind of thinking. That is why systems design is so difficult to teach. In this chapter, we can tell you what analysts do and how they use analytical tools during the design process, but can not explain how analysts conceive new systems. Design methodologies and techniques are merely part of a tool kit that analysts use to practice their craft. To arrive at a system's design, analysts draw upon their knowledge of information technology, their experience with computer systems, and the information they collect about the problem at hand, the environment in which the new system will operate, and the expectation of management and end-users. They then synthesize this information. The design evolves from observing, modeling, and thinking about the problem. Through experience, education, and professional and personal development, analysts hone their design skill.

3.2. Structured Design as process

3.2.1 The phases of design

As discussed above, there are various techniques and tools being used in the analysis and design phases, however, not all of them are suitable for any systems. Even when a modelization technic is employed, there are stil different levels of complication of that model in different cases. If too few system development tools are used, the system's quality will be worsen, and vice versa, if more tools are used than they are actually

needed, resources are wasted. Thus, analysts and designers have to judge to give the right decision of what techniques and tools should be used and to what extent.

The recommended percentage distribution between analysis, design, and implementation is 50%-60% for the analysis and design phases and 40%-50% for the implementation phase.

There are three primary design stages:

- Initiation: This is the form the design takes when it has first been generated. It is usually not complete, in that some functions overlooked in the analysis may still be missing, error processing is missing, and details of how outputs are produced will also be missing.
- Abstraction: Derived directly from the initiation results, this design phase involves the completion of the design. All functions and necessary detail missing in the abstraction phase are incorporated into the design. This is the phase in which we apply coupling and cohesion, obtaining the best design we can given the constraints of time and talent.
- Instantiation: Based on the abstract design, this phase involves the revision of the design to make use of specialized operating system features, the use of hardware features rather than software to provide needed services, and the incorporation of any additional functions or features that have been identified as being needed. During this stage the software engineer makes rational compromises with respect to coupling and cohesion in order to attain some performance goal or to meet some external constraint.

These stages are organized into two phases:

Logical Design: The objective of this phase is the development of a design which is directed by an abstract machine and operating environment. This model contains the highest amount of quality and quality components as we were able to incorporate, with the time and talent available.

Physical Design: The objective of this phase is the creation of a blueprint of the system. This blueprint is complete and correct and accurately describes every aspect of the system to be developed.

Model of Structured Design is composed of two distinct but related phases. In chronological order, the first is logical design phase and the second the physical design phase.

Logical Design Phase

The activities which occur during the Logical design phase are the following:

- Develop the preliminary design: the goal of this task is to create a conceptual design which will reflect in the analysis package and contain the amount of quality that we can incorporate into it
- Define the man-machine and machine-machine boundaries on DFD: Using the level -'0' dataflow diagram developed during the logical modeling phase of analysis, identify which processes or functions will be allocated to hardware, software and manual procedures.
- Transform E-R-A Model into Relational Model: Each entity and each relation is transformed into a logical relationship using the Relational Database Model notation and semantics. The relations are intergrated into global relational schema. An E-R-A Model is a conceptual representaion of real world, enterprise objects. It is composed of entities (with their attributes) and relationships between them.
- Normalize Relational Model: the logical relational schema is transformed into a normalized schema to provide for logical data access and update intergrity.
- Prepare Database access Model: For each user view a logical data access specification is created, which defines the process followed in accessing the required data and the logical relations involved.
- Verify against Normalized Relational Model: Inability to support required data access will cause the E-R-A Model and the logical relational Model to be refined appropriately.
- Transform leveled DFDs into first-cut structure chart
- Refine structure Chart using coupling, cohesion and adding additional functions, as necessary.
- Develop pseudocode: the pseudocode for each module describes what the module does. The basis for most pseudocode is the structured English that was created during the analysis phase.
- Revise Event Model: Refinement of this model is necessary in order to put it into a more acceptable form to accommodate the design and implementation perspectives.
- Revise Relational Model: Bring this view of the system into compliance with the other revisions and refinements that have taken place.
- Update Lifecycle Dictionary: One of the outcomes of transforming the analysis results into an initial design model is that new data elements have to be incorporated. Since Lifecycle Dictionary is the repository for information, it must be updated to reflect new parts of systems, descriptions of each of the modules.

- Perform consistency check among Structure Charts, pseudocode, Event Model, E-R diagrams, Relational Model. Ensure that all parts of the design are accurate and mutually supportive.

Physical Design Phase: has tasks as followings

- Develop Detailed Design: It is derived from the logical Design by incorporating considerations of system size, timing constraint, the existence of hardware or software utilities, language features and other implementation considerations.
- Review Preliminary Design Package: The purpose of this task is twofold. One is to identify and correct errors and necessary refinements. The other is to ensure that all that is necessary to accomplish the physicalization of the design is present and of sufficient quality to support this next phase.
- Develop final ('build to') Structure Chart
- Update Lifecycle Dictionary to reflect physical characteristics: The lifecycle Dictionary must include the compromises between the desired (Detail Design) system and the obtainable (Physical Design) one. The primary inputs or changes at this stage are: the 'build to' Structure chart process, dataflow and data access specifications; the implementation constrains from the packaging criteria; the changes to the Event Model; and the new DBMS data model and its operational characteristics.
- Revise pseudocode: update the content of the pseudocode, as necessary, in order to accommodate the changes made to the system design during the packaging process.
- Perform coupling and cohesion analysis
- Revise Event Model
- Reconcile all elements of modul design pakage
- Map to DBMS Data Model: The logical, normalized schema is translated into an initial DBMS schema. This schema is then refined to take advantage of the unique capabilities of the DBMS being used. The translation process is specific to the particular DBMS Data Model. Each translation starts with a logical relation and converts it into an appropriate DBMS structure.
- Optimize DBMS schema access costs: Cost models are to determine optimal access paths to support the user daa view requirements. Any modifications to the DBMS schema are verified against the logical relational schema.

- Revise implementation estimates: This supports the concepts of iterative estimation refinement that is used in most engineering fields.

Example:

From the above library management case study, we realize that the system analysis stage focuses on the logical aspect of the system. It does not consider at all the implementations of the requirements. By contrast, the system design phase immediately considers the ways to set up the professional requirements by using computer.

The inputs of system design phase are the specific requirements which were built in the system analysis phase, including:

- Function Hierarchy Diagram
- Data Flow Diagram
- Entity Relationship Diagram

In the system design phase, we have to:

1. Define which functions have to be done by human, and which functions have to be carried out by computer, so we have DFD system (half-physical)
2. Design database
3. Design the human/computer interface

Unlike the system analysis phase which needs a few tools and technique, mainly for structure building, in the design phase, the choose of using a tool is very flexible. Some tools are used in detail and closely, some other are used more flexibly, and some may never be used. It depends on the requirement of the problem and the experience of the system designer.

With the library management system, the system design phase concentrates on the following main parts: to determine the computer system, to design modules and to design the database.

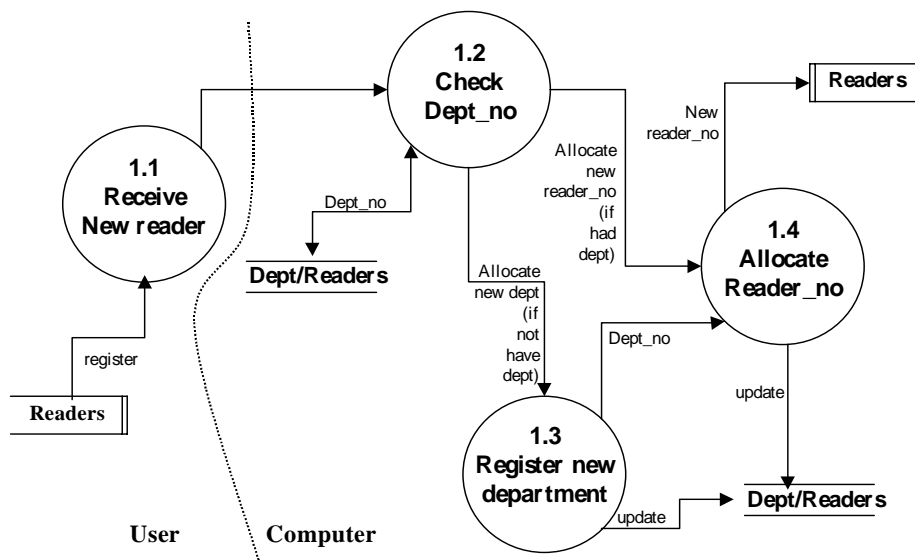
3.2.2 Identification of the computer system

This is the first phase in system design and it should define which part of the system should be handled by computers and which part by users. In this phase, business DFD will be used in both requirement description and in other processes at the lowest level and consider the role of computer in each process. Looking at business DFD as a logical model essential to the system, system DFD can be better described as “half-

physical”. This is because although it has formed the processes to be handled by computer, it hasn’t expressed the way to implement them, the creation and usage of programs and data files. Basically, the processes of the system DFD are built on the basis of logical processes of relevant business DFD. In fact, if one of the business processes is transferred to the designing phase or as a computerised or non-computerised task, it is still named the same in both business DFD and system DFD models. In case of a logical process is designed to be partially computerised and partially manually implemented , it is required that any new processes in the system DFD must be traced in the processes of the business DFD from which we will make the apart later.

Determines the computer system in the presented example:

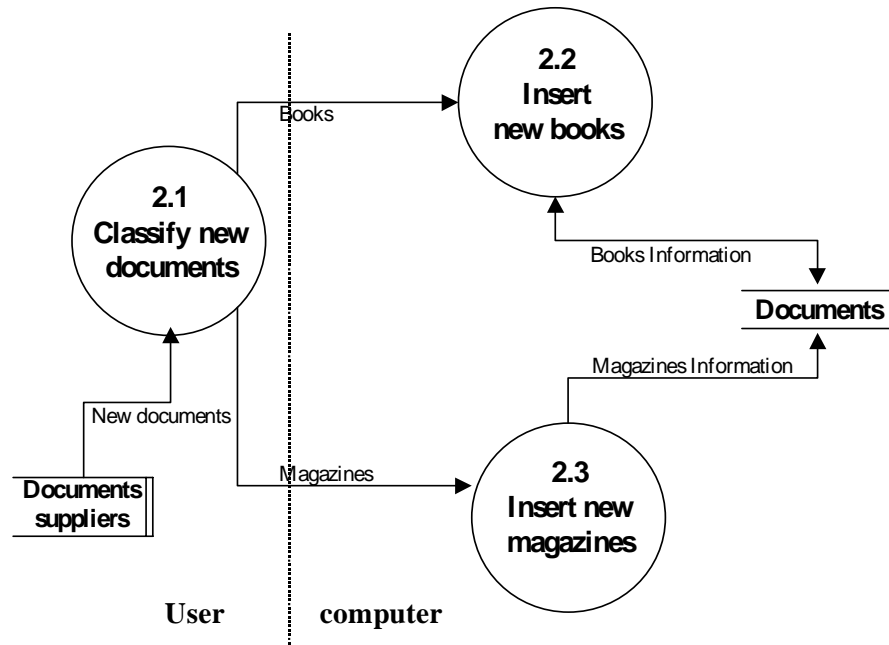
This is the first stage in the system design phase. Depend on the DFD business in the system analysis phase, we build the DFD system to determine which functions to be carried out by computer and which functions to be done by human:



Data flow diagram (Function 1).

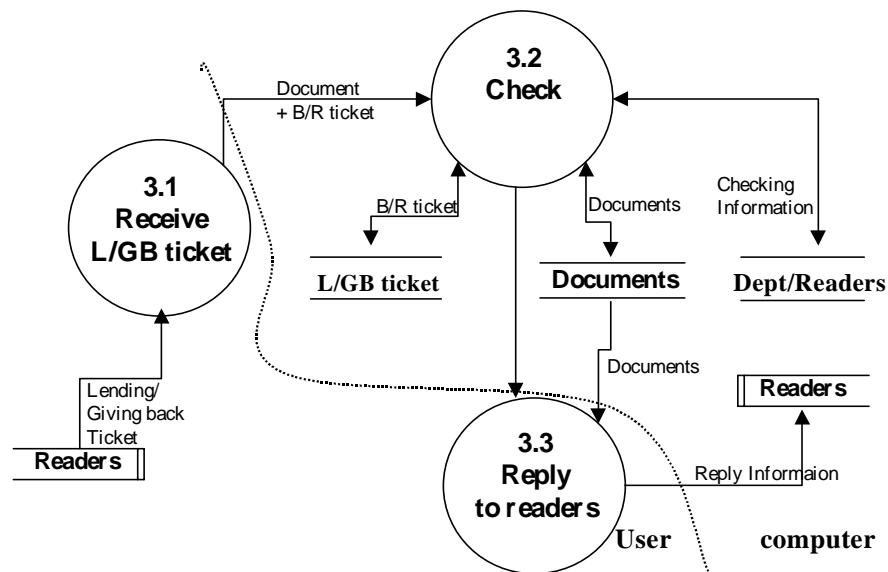
In this Reader management function, we realize that when a new staff wants to be a reader, he (or she) has to register with the librarian, so the Receive New Reader

sub_function has to be done by the librarian. These other sub_functions will be done by computer.



Data flow diagram (Function 2).

In the document management function, we realize that new documents must be received and classified by the librarian. After that, the librarian will use computer to input information of document which have just been classified as books or magazines. Therefore, we have the above diagram.



Data flow diagram (Function 3).

In the Lending/Giving back document ticket management function, we realize that there are two sub_functions: Receiving Lending/Giving back ticket and replying to readers must be done by the librarian, the computer only takes on sub_functions: checking the input information about these ticket whether correct or not and returns the result for librarian on the computer screen. So we have the above diagram.

The two remaining functions namely: Looking up documents and reporting have to be carried out by the computer, because in fact that the processing documents are stored in computer now.

3.2.3. Model Construction

Constructing the system data flow means outlining the system designer's ideas on how computers should be used in the system in the future. DFD, which assists the analysis and design process, helps designer modelize his idea quickly. On that basis, discussion among system developers are held to fine-tune the model.

3.3. User-computer interface design

3.3.1. Dialogue design

Interface designs: There are various types of user-computer interface designs, each of which has a typical character and ability. The design type is required to be suitable to the system's duties and to its users who will interact directly with the computers

Significant criteria for the evaluation of dialogue type:

- Easy to use: easy even with inexperienced users
- Easy to learn: easy for users to remember
- Processing and responding speed
- Easy to develop
- Significant types of user-computer interface design:
 - Q&A: Questions or pop-up reminders on computer are by turn answered by users. This type of design is simple and suitable with inexperienced users
 - Menu table: Options are classified and displayed on the screen. Menu table is frequently used as a mechanism for connecting to the system. It is a good approach if the screen displays a full menu. This type of design is suitable with inexperienced users but boring with more experienced users
 - Symbols: Symbols are displayed on the screen to stand for various functions. They are easy to learn and they enable fast access. In fact, graphics occupy more space on the screen and they are not as economic as menu table. The main disadvantage of symbols is their inability to describe the options lively, clearly and meaningfully. In order to develop symbol-dialogue, professional softwares are necessary.
 - Form: Filling in the form is a popular type of dialogue on data and data processing. Forms are displayed on the screen similarly to the way tables are arranged. The screen also displays form name, field name and instruction information. The pointer controlled by a software moves automatically among fields or by using TAB or carriage return – enter keys. The advantage of form is its close contact with users. This type of design is suitable to all users.
 - Language command: This is a wide but simple area consisting of both simple commands and grammatically complicated commands. A command will result in a move of the system when it is entered by the user. The most significant advantage of language command is that its flexibility is limited by the language's grammar only. However, it takes time for users to learn by heart the commands and users are required to have a background knowledge of the system in case there are no information displayed on the

screen. Language command asks for great efforts while developing it. It is suitable for users who are professionals.

Essential instructions in dialogue design

Feedback information: provide users with the information on what are being done

Status: keep users informed of the system's parts they are using

Escape: allow users to exit from one manipulation

Minimum tasks: Avoid users from making too many manipulations

Default: Set the frequently used parameter

Support: Provide users with necessary supporting information

Cancel: Users can cancel and resume

Consistence: The implementation of commands must be consistent via interface

3.3.2 Screen design

In this process, the most important thing is that the displayed information, command, notice of the systems status go in line with each other and are arranged in priority order in particular cases and comfortable to users. There are 3 main types of display:

- Menu display: which helps users in fast connecting and easy access to system's functions
- Dialogue display: that consists of notices/dialogues between users and the system
- Data entry display: which organizes data in groups of information classified by changability, frequency of use, importance. Depending on the situations, designers will decide whether to design simple or master-detail data entry display

3.3.3. Outputs Design

A number of basic design principles ensure that the output is presented in a way that is easy to understand and interpret. They are as follows:

- ~ Notes, headings, and output formats should be standardized whenever possible. Format consistency is an attribute of 'user-friendly' output. Users feel comfortable. With familiar layouts.
- ~ The arrangement of information should be logical. Information should be presented in 'chunks' or quantities that can be easily absorbed in language that is easy to understand.

- ~ Acronyms and abbreviations in output should be avoided especially when the output will serve novice users. Define words that may be unfamiliar to the user.
- ~ Algorithms and assumptions on which calculations are based should be available to users of the output. This assures correct interpretation of output.
- ~ The user should be able to locate needed data quickly without having to search through all of the data.

In this library management system, the user_computer interface will be designed in the type of menu table: all first choices will be displayed on the monitor, and the next choices will be display in the hierarchy menu table as suggestions. Each choice has a hot key which is underlined:

<u>S</u>ystem list	<u>D</u>ocument	<u>L</u>ook up	<u>L</u>endGive <u>b</u>ack<u>T</u>icket	<u>R</u>eport	<u>H</u>elp
<u>D</u> ePARTMENT	<u>B</u> ook	<u>B</u> ook	<u>B</u> ook lend	<u>B</u> ook status	<u>T</u> opic
<u>R</u> eaders	<u>M</u> agazine	<u>M</u> agazine	<u>B</u> ook giveback	<u>M</u> agazine status	
<u>N</u> ation			<u>M</u> agazine lend	<u>D</u> ePARTMENT	
<u>L</u> anguage			<u>M</u> agazine give back	<u>O</u> verdue reader	

Speciality

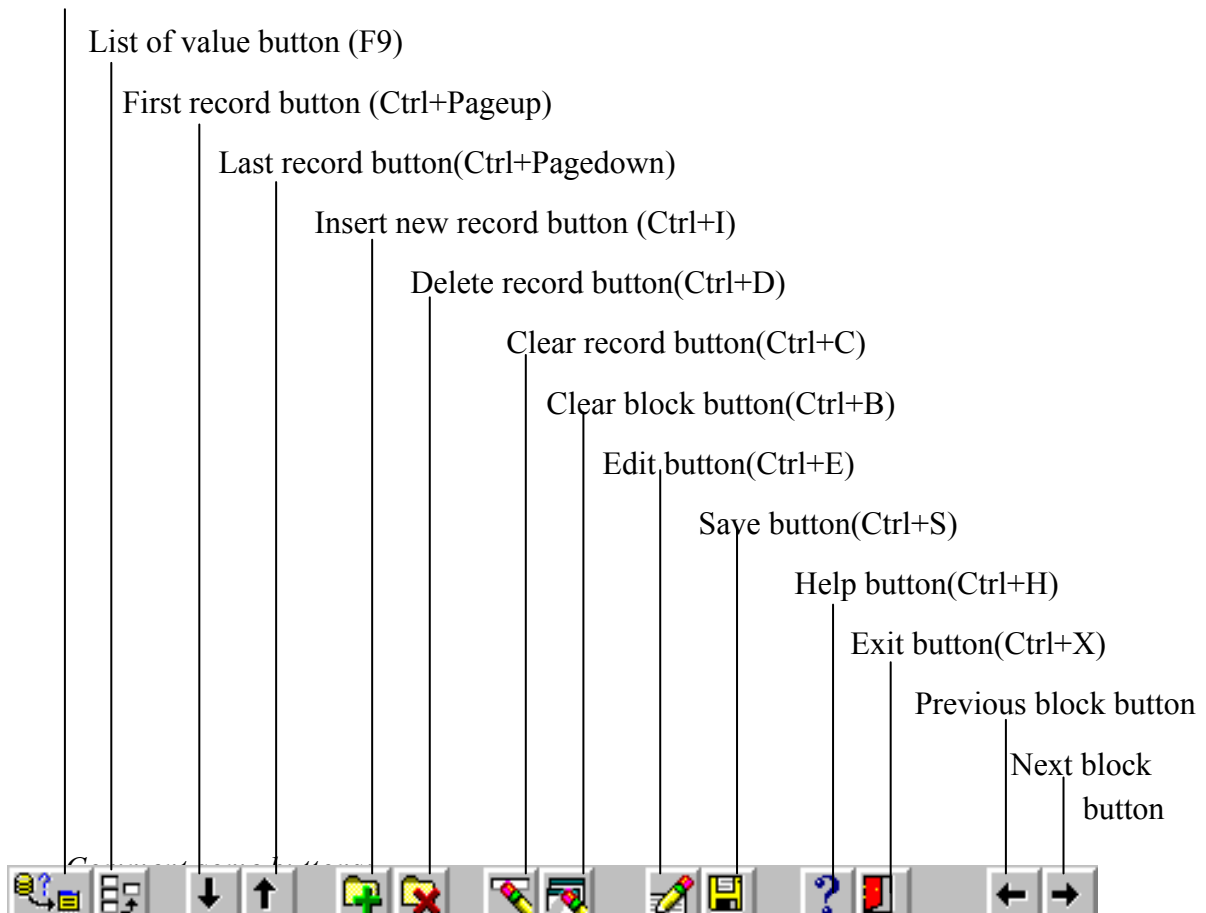
Collection

Exit

Screen design:

General requirement of modules: Modules must have a common toolbar, including the following buttons:

Query data button (F8)



Query data button: used for looking up record which satisfies the value entry.

- ~ List of value button: used for listing all values that user can choose among them, notice that only some fields can have this function.
- ~ First record button: go to the first record rapidly.
- ~ Last record button: go to the last record rapidly.
- ~ Help button: Display help page about current form.
- ~ Next block button: If there are more than 2 blocks in the current form, this button will go to the next block.
- ~ Previous block button: If there are more than 2 blocks in the current form, this button will go to the previous block.

For data entry: organize manipulation of data tables and organize data in groups of information classified by changability, frequency of use, importance, depending on the situations, and group them in a block. In each data entry, comments should be shown, and by pressing the Tab button will move between data entries.

The form modules follow:

1. Department:

Form module for entering the list of department in the institute, include number and name of departments:

order	table_name	data privilege
1.	DEPARTMENT	SELECT INSERT DELETE UPDATE

2. Reader:

Form module for entering the list of readers in the departments of the institute, including number, name, address, birth_date, comment and the number of department that the staff belongs to.

order	table_name	data privilege
1.	DEPARTMENT	SELECT
2.	READER	SELECT INSERT DELETE UPDATE

3. Nation:

Form module for entering the list of nations including number, ISO name and Vietnamese name of the nations.

order	table_name	data privilege
1.	NATION	SELECT INSERT DELETE UPDATE

4. Language:

Form module for entering the list of languages including number, ISO name, Vietnamese name and the system of languages.

order	table_name	data privilege
1.	LANGUAGE	SELECT INSERT DELETE UPDATE

5. Speciality:

Form module for entering the list of Speciality including number, and name of specialities in the library.

order	table_name	data privilege
1.	SPECIALITY	SELECT INSERT

		DELETE UPDATE
--	--	------------------

6. Collection:

Form module for entering the list of Collection including number, and name of collections in the library.

order	table_name	data privilege
1.	COLLECTION	SELECT INSERT DELETE UPDATE

7. Book:

Form module for entering all information about books such as: number, name, keywords, number of collection, number of specialty,.... In this form, the number of book will be created automatically as the system analysis phase indicated.

order	table_name	data privilege
1.	NATION	SELECT
2.	LANGUAGE	SELECT
3.	SPECIALITY	SELECT
4.	COLLECTION	SELECT
5.	BOOK	SELECT INSERT DELETE UPDATE

8. Magazine:

Form module for entering information about the magazines will be designed in master-detail entry display, with the master display for information about all kinds of magazine, and the detail display for information about each kind of magazine such as: year, month, volume, number, quantity.

order	table_name	data privilege
1.	NATION	SELECT
2.	LANGUAGE	SELECT
3.	SPECIALITY	SELECT
4.	COLLECTION	SELECT
5.	MAGAZINE_HEADER	SELECT INSERT DELETE UPDATE
6.	MAGAZINE_DETAIL	SELECT INSERT DELETE UPDATE

9. Book lend:

This form is designed in the form of lending ticket including: number of reader, number of book, the lending day, the day will give back, and comment.

order	table_name	data privilege
1.	READER	SELECT
2.	BOOK	SELECT
3.	L_GB_BOOK_TICKET	SELECT INSERT DELETE

		UPDATE
--	--	--------

10. Book giveback:

This form will be designed in master-detail entry display, with the master display is the list of readers who are lending books in the library, and the detail display is the list of lending books of each reader. If the reader gives back a book, the librarian will enter the current day into the giving back day. And that book will be set free.

order	table_name	data privilege
1.	READER	SELECT
2.	BOOK	SELECT
3.	L_GB_BOOK_TICKE T	SELECT INSERT DELETE UPDATE

The form of lending and giving back magazines are designed like the form of lending and giving back books, except that is added the following fields: the year, month, volume, number of the magazine.

Design the looking up document function:

The searching activity depends on the name of author or the name of book. It is carry out by deciding a string whether is within another string or not.

To make it clearer, we consider this example:

If the reader only remembers the name of author beginning with e.g the `b` letter, he (or she) can enter `b` into the name of author field, then press Look up button. The looking up function will list all books of these authors (who have books in library now):

F.Baccelli and G.Fayolle

THOMAS C.BARTEE

Bui Van Sinh

Nguyen Van Binh

Or if the reader only remembers one word of the name of the book beginning with string `conf`, he (or she) can enter this string in the name of the book field, and press Look up button, The looking up function will list all books which have one word beginning with `conf` string, as follows:

AFIPS Conference Proceedings

AIAA Guidance and control conference

Confidence

All the books found will be listed in a block including the number and the name of books. If the reader wants to know more information about one book, he (or she) can focus on that book, then more information will be displayed in another block. This design is very suitable and convenient for readers.

The looking up magazine function is designed in a form module like the looking up book function, but it adds more information about the year, month, volume, and number of magazine.

Design output:

The output of the library management system is 3 main reports

- Report about document: which includes books and magazines
- Report about lending and giving back documents of departments: which displays information about the lending and gives back documents of all departments in the institute.
- Report about overdue readers: which lists all readers who are now overdue.

All reports can be printed in the A4 size, or be displayed on the screen or even to be stored in a text file. Particularly with the report about the status of documents in the library, it can be listed in a specific time.

3.4. System monitoring design

In order to ensure that the system will operate efficiently and safely, tests are necessary at different phases of the system development. Not only the analyst, but also the management, users, observers or those responsible for managing the system afterwards should get involved in control, research and analysis. Control has been regarded as a boring task which goes through a series of boring testing items. However, if we have a good approach and are serious about it, this is one of the most important thing to do during the development of the system. At this point the analyst predicts any potential loss due to the low quality of information provided during the processing phase and makes decisions for the system to minimize the loss. In fact, the following significant aspects of the system should be protected by control:

- ~ Accuracy: Tests are required to ensure the accuracy of system's activities and information filed in the database
- ~ Safety: Access decentralization is needed among users (should be to the different functional levels of the system and access traces should be recorded via the password mechanism), so as to prevent the system from being illegally accessed. At the same time, information management measures are necessary to avoid information losses
- ~ Privacy: Users' access authorization needs to be protected
- ~ The most popular method of analyzing controls is based on the dataflow diagram (DFD) whereby the analyst will go through various DFD models, figure out the roots of weaknesses that may need control. We'd better focus on those controls relating to the defined system. The method of analyzing controls consist of the following phases:
 - ã Define the unprotected points of the system: The unprotected points are those points where information can be accessed by unexpected users. The unprotected points can be the input and output ends, the display of users-computer interface, data warehouse or files. We can go back through the DFD to check every processes and dataflows leading to the unprotected points
 - ã Design necessary controls

After defining the level of loss that may incur at the unprotected points, the designer has to decide whether to take physical controls to fight against or minimize the loss. However, the use of control means a change in the corresponding designing models and DFD

Thus, after the analyst has finished researching, the details of necessary controls for the system will be handed to other relating analysts who will organize the controls into

relevant processes and then submit them to users as control samples. Ideally, both analyst and users should attend discussions with the controlling group. In this way, users can install and accept the control type that has been decided in an easier way. However, this is not always feasible because there are not many people understanding the details of the system's operation. Thus, the most important thing is that effective control is put in the right place and users are aware of how the control works.

3.5. Organizing the system's components

The system is made of different subsystems, each of which consists of various computer programs. The program itself is made of various modules, each of which is designed to handle a function inside the program. At this phase of the system design we are only interested in functional modules of the program presented as processes in the system DFD and related non-computer processes. Now we need to consider the way to group the modules into a program and the best way to organize the program in a sub-system.

3.5.1 Grouping criteria

According to entity group

The largest components of a modern system is referred to as subsystems and each subsystem is formed by one of the system entity groups. Each entity group is a group of all entity types relating to an importation communication processed in the system. After arranging the main entity types of the system into the most relevant entity group, the designer can list all functional modules of the program to decide whether he should form, modify or delete an entity type in a group. These modules are considered "candidates" of a subsystem. It should be noted that some modules can be considered to be used in different entity groups

According to events

The relationship between the modules over time should be considered. If each external event asks for the implementation of several modules or the implementation of this module requires the implementation of others, it is necessary to put them together into a program.

According to accuracy

There are other criteria used to decide the way to better group the system components together, however, the use of this criteria or the other depends a lot on the circumstance of the organization (the way to do business, existing hardware, staff's skills). There are 3 general situations where the grouping of modules and programs are possible:

- ~ Modules are already put in blocks online and appear in the same area, or have the same relationship with a type of data
- ~ Users in different responsibility levels working on the same data
- ~ Each user has different responsibilities relating to various functions

3.5.2 Designing approach

The objective of the first phase is to put the modules together into programs and sub-systems and to discuss the criteria to do so. From this point on, you'd better go up from the bottom to decide which module should be grouped together to make up a program. Then give the programs relevant names and consider the relationship between them. It should be noted that during this phase, designers are required to thoroughly study their grouping proposal and organize the system in another structure. When the designer identifies a process in system DFD which is not put into the system, he tends to find a suitable place for the missing process. However, this may result in a reconsideration of the whole solution proposed. The system design is often organized as a hierarchy. Modules are grouped into programs, programs put together into sub-systems, and sub-systems form the whole system.

There are 2 module types that need to put into the program being designed: functional module and linking module

- The Functional modules are used to perform particular functions, most of the processes presented in the system DFD belong to functional modules
- The Linking modules are not used to perform any particular functions, they are supposed to offer an interface between functional modules and control the accessing to these functions by users.

It is also important in this phase that although most of the modules are unique and in connection with a defined situation, some others have various functions and can be put

into different programs and subsystems. They are referred to as “sub-set” and should be processed in a different way from special program modules.

From the very first phase of designing, the analyst has to patrol through the system to identify opportunities to use available “sub-set” or build them up from scratch. Linking modules are often used in the phase of “linking the system base on the computer”, when the system’s functional modules are grouped into the actual program. Designers normally have to create an interface for users to select the necessary functions or create a dialogue asking for password so that user’s access to different functions of the program will be censored.

3.5.3 Modelization tools and techniques

The two models popularly used in this phase are the System Diagram and Computer Dataflow Diagram. In fact, these techniques that help simplify the technical description of BFD and DFD have been discussed in the business analysis phase

a. System Diagram based on computer

This is a very simple form of model which illustrate the functions of the system base on business functions described in BFD in the analysis phase. It also points out the interdependence among modules, program, subsystems and the whole system. This is a hierachy diagram using the same symbols and standards as BFD does.

There are often 2 levels for system diagram of a system. The upper level is a page consisting of all components including “program”, the lower level are different pages for different programs, it points out the modules dependent to the programs. However, larger system may require a medium level where contents of each subsystem are illustrated in separate pages.

Although the rules to name the subsystems and programs are flexible, you’d better use those names that are short, unique, reminding and connecting to the words “program” or “subsystem”. Functional modules should be named the same as when they were defined. This is important because it helps analysts to form the relationship between modules and their initial business point of view, especially changes are to be made to the system.

Components of the system should be numbered whereby each program is given a unique number. This helps identify which system a subsystem belongs to. However, if each functional modules can still keep the logical symbols given to them in the business analysis phase, the phases to come should be easier.

It should also be noted that linking modules are not described clearly in the system diagram. Assume that every program has a linking module responsible for controlling the access to functional modules, the linking module can be considered to be presented as a block with a program name in the upper level. In the case where there are needs to insert more linking modules, we can use a block describing this group as “sub-program” in the hierarchy order. System diagram is a very important model, it not only gives us a panorama of the whole system but also provides essential material for the analyst, the maintainer and the system developer during its operation period.

b. Computer Dataflow diagram (DFD)

This is a main model to describe the system’s structure. It is built up based on dataflow diagram modelization with symbols having the same meaning as described in the last chapter. Computer DFD has the same relationship with computer system diagram as functional DFD’s relationship with business functional diagram (BFD). In other words, these two diagrams must be fully consistent and this one is used to test, modify and improve the other. The most significant difference between computer DFD and system DFD is that the non-computer process is presented in system DFD. This shows that although we are not interested in these processes as part of the “computer system”, we still need to figure them out as a source and recipient of inputs and output. In the lowest level of computer DFD, inner effects are named by the process pointed out to users in system DFD, however, they must be marked by name of the departments inside the organization.

It is essential to display the linking modules in computer DFD. This may make the module more complicated than it should be in reality, because linking modules must be connected to by inner effects so that the connection to relevant functional modules can be made. However, once this connection mechanism is set up, there will be direct dialogue between inner effects and functional modules, meaning that two more information exchange flows are needed to modelize every user-computer dialogue.

The files presented in computer DFD resemble the files in system DFD, however, in this model, they are presented as physical files which are planned for use in the system. This means that the files illustrated in computer DFD can be assumed files of the sample, or database structure at a high level.

The concept of “outer effect” in the computer DFD is used to demonstrate the idea whereby a file of a computer is shared with other computers in the system. For those shared files, there is a principle that they can be accessed to from other systems and they must appear at the top of the diagram. In this diagram, other systems are displayed by outer effect symbols, and the dotted line is used to link the outer effects directly to relevant

data files. It should be noted that the dotted line here is not a dataflow, it is only used to present the files also be used by other systems. The arrow direction shows the ownership relationship of the shared files.

Thus, in order to have a clearly displayed computer DFD, we can locate the components of the diagram in the following order:

- ~ Inner and outer effects (if any) on top of the diagram
- ~ Intersection modules
- ~ Functional modules
- ~ Relating files

This arrangement can minimize the number of overlapping data

3.6. Analysis of data usage and logical navigation

Before designing the physical database, we need to check the data model and the ability to meet the detail requirement of the computer system. This is done by analyzing logical paths. Logical path analysis is defined as a process testing the relevance of data model and providing detailed information of data access requirements for designing physical database. The 3 main tools used in the area are:

- The path analysis diagram
- The navigation model
- The data usage schema

3.6.1 Path analysis diagram

The path analysis diagram shows the order of accessing different entity tables. It figures out the frequency and nature of different accesses. Designers have to point out the trail from inputs to outputs which has been defined via the relationship and entity types of the data model. Each path presented in the path analysis diagram has to correspond to the dataflow in data model, regardless the relationship direction is one-many or 'many-one'. However, it should be reminded that the transformation of relationship from "many" to "one" will result in losses of information characteristics, because information about the "one" case is often more panoramic than the one in the "many" case.

The logical path analysis diagram has the following main purposes:

~ To check whether the necessary information for the process can be achieved by using data model

~ To describe the complication of the process and the data model already built, whereby assisting physical database designer to optimize his data structure.

In some cases, the logical path analysis diagram is used as a program-designing tool relying on paths of a superior program's block diagram. The advantage of logical path analysis diagram against the traditional block diagram is its construction is based on the structure of data used in the program module it refers to. However, building up a logical path analysis diagram is not always necessary (for example in the case of developing small systems)

3.6.2 Navigation model

As discussed before, the path analysis model gives us an idea of using data in all processes of the system. The navigation model will give an overview of data usage throughout the system; it gathers information of individual access paths in different processes and links them to the data model already built in the analysis phase.

The navigation model is a data model of the system with individual access paths added. The access paths are illustrated in the model, by using broken arrowhead lines to show the path direction. The numbers beside the line tell that the paths are connected together. The access to input points is also recorded using the same symbol in the path analysis diagram. The navigation model maker will begin at the top of the path analysis diagram, then going along the accessing points and work out down to the bottom. This maker will also compare the entity types with the types in the model, and marks the path direction from "one" to "many" or from "many" to "one". After the model has been finished, tests are also necessary for possible shortcomings and irrelevance. In case there is a relationship that is unused, the following 2 reasons should be considered:

- The relationship is unimportant and can be taken out while building up the physical database structure
- Some important processes have been missed or path analysis diagrams have not been gathered adequately.

However, how to use this tool or another and how to combine the tools for the system development is also dependent on actual conditions:

- When the system is developed in a small scale, and the effectiveness of files is not of leading importance, the designer can show the path of any process on the data model and omit the path analysis diagram. With too simple systems, the designer doesn't even need to put the name of the paths on the model, and the only needs to point out which relationships are used.
- When the system is developed in a larger and more complicated scale, the path analysis diagram is inevitable. However, when these paths refer to the same data model, the overall Navigation diagram may become too enumerated to follow. In that case, we should split the navigation model into different levels corresponding to the processes of the data model (for instance, each model is corresponding to a subsystem). Nevertheless, there are possibilities that information is overlapped in different models.

3.6.3 Data usage schema

This chart contains the details of each entity type that get accessed in the model, the number of entities that may be accessed to, the frequency, the features used for access, and other aspects assisting designers to build up the database.

In general, the analyzing process of data usage can be started right when the very first requirements of the computer system is processed and is continued through the following development steps of the system. Building model and data usage chart are set up to tell all entity types the access in which they may be required to participate. Changing from viewing data as separated processes in the model to keeping track of data accessing status of entity types and frequency is significant to physical database design.

3.7. Design of Databases

3.7.1 The common problem of database design

Data requirements

The raw material of information systems is input data, i.e. the representation of facts or concepts that becomes usable when processed.

Once output is designed, data elements required to produce this output can be ascertained by a process of deduction. For example, the overdue book borrowing readers report.

There are two ways for an analyst to determine the data requirements. The first is charting. First, the analyst lists processing goals. Then, each of these goals is subdivided. The input can be deduced by using design tool such as Structure chart (HIPO, etc) for each output.

Another method commonly used to derive data element requirement is an Input/Output table (I/O table). The table has a horizontal axis listing output reports and files. The vertical axis names data elements that analyst identifies as necessary for these reports. The analyst has to mark the cells to show what data element are used in what reports.

Some of the required data elements may already be found in databases of the organization and defined in data element dictionaries. If not, data element dictionary entries must be prepared and the collection of all data element values must be planned.

The analyst is now ready to organize the data that the new information system will use.

Data organization by files

File systems join data elements that are logically related into records. The values of these data elements, collected during implementation of the system's design, will be stored on a physical medium such as tape or disk.

Once records are logically designed, the analysts group related records into logical files. Theoretically, there is no limit to the number of data element in a record, and the number of records in a file. During the design phase of the new information system, a check will be made to examine whether a file containing the set of data element needed to support the application exists. If not, it may be necessary to create a new file. A single file may support more than one application, and that new application may result in the creation of new file.

File linkage:

Sometimes a single application may process data stored in several files. In order to access this data, the files must be linked by a data element that is common to two or more files. Linkage allows data to be shared and used for multiple purposes. A collection of linked file is called an *integrated file system*.

Problems with File Systems:

When a company has many files as many organizations do, the problems inherent in file systems become acute. There is high level of data redundancy since many data elements are stored in more than one file. This duplication is costly in terms of data preparation, storage space, and processing time.

Furthermore, file maintenance becomes an expensive proposition. Every time the value of given data element changes, that change must be made to every record in which the old values occurs. Too often, the change is made to only one record. This means that a specific data element may have different values in different files. End-users do not know which report to trust and begin to doubt the accuracy of all computer output.

File systems lack flexibility. To answer a request for information may require analysts to restructure existing files. Should a record be modified to meet the information request, every application program in existence that uses the same record must also be modified. In the file environment, one change can trigger the need to make other changes so that even trivial modification can be time consuming and costly.

The data base approach

A different concept of data organization, the *database approach*, is today replacing file systems in many medium to large-size companies. Data base systems have a single data pool that is shared by many users and applications. There is only one representation of each data item within this shared data pool. This avoids redundancy of storage and data inconsistencies since the value of each data element is generated, validated and stored just one. Instead of scattered computer files controlled by many different users, responsibility for data in data base system is assigned to a data expert called Data Base Administrator (DBA)

A specialized set of programs, called a Data Base Management System (DBMS) is generally acquired to help computer users access and manipulate the data in a data base system. Most DBMSs include a special query language that allow users to access the data base, retrieve information from data base without having to write a program. Data can be added, retrieved, updated, and deleted from data base using DBMS functions. If writing an application program for a transactional system n batch, the programmer accesses and uses the data base by writing the program in a record-level language provided by the DBMS, which is called the *host language*.

An important function provided by a DBMS is *data independence*. Data independence means that users and programmers are insulated from database. They do not know how the physical database is stored.

Since all access to data is made via the DBMS, data base management systems can perform another important function- *data access protection*. For example, the DBMS might be programmed to refer to an internal security matrix to determine which data

elements a particular user is permitted to access and what type of operations that users can perform- read only, write only, or read, write and delete.

Logical organization of data

We have discussed the features and limitations of the two types of data organized in-file systems and data base systems. Let us now look at how systems analysts plan the logical organization of data element for each of these systems.

For file system, the organization of data elements into records, and records into files is relatively straightforward. Analysts study diagrams of the data flow and other analysis documentation that has been prepared during the analysis phase. Then they decide how data elements relate to one another and how records should be grouped in files called logical schema of data. They may perform draw structure diagrams, or use model such as the Entity-Relationship (E-R) Model. The logical schema of the data serve as bases for the physical storage of data value on tape, disk...so that the data can be efficiently accessed by application programs.

The logical organization for data base system is much more complicated. The size of the data pool and the complexity of relationship between data elements in that pool make it difficult to conceptualize an appropriate organizational structure. Analysts choose one of three common data models for logical organization-hierarchical, network and relational model. The analyst must have a thorough understanding of the database under development (and the relationship among data elements in that database) to choose an appropriate model. This understanding is acquired by studying data requirements of the new system and using data base design techniques to analyze and sketch the data relationships.

To understand the term 'data relationship', some data model terminology must be learned. An entity is what users have expressed interest in recording data about. An information under development will process data about attribute of these entities. Each entity has a value, which provides information about the entity.

The question that analysts must resolve is how do entities and attributes relate to one another in a database. Three types of data relationship can be identified

- One-to-one (1:1)
- One-to-Many (1:M)
- Many-to-Many (M:M)

Hierarchical Data Model

Most of you are familiar with this hierarchical structure. This same inverted tree structure of nodes and branches can be used in the organization of a database. In this mode, a node (representing a file, record, and entity...) with dependent node is called a parent, while each subordinate node is called a child. Note that this parent-child structure is characterized by one-to-many relationship since each node is pointed to by only one node above it. When searching for data in this hierarchical model, every node has to be accessed through its parent (except the root).

With this hierarchical parent-child structure, some of the data may be inaccessible with a poorly structured database. When this occurs, it can be a formidable task to restructure the database in order to access this data sufficiently. Another problem is that new links must be forged when new kinds of data are added to database. Care must be taken to ensure that these new links do not invalidate or corrupt existing connections.

Network Data Model

The network data model resembles the hierarchical data model with one important exception—a child can have more than one parent. Note that there are lateral connections as well as the top-down connections within the database. The complex structure of relationships complicates data base design and modification, but the model offers flexibility in searching for data without much loss of speech.

The network data model has not become a standard data model because it is highly complex and incohesive. There are many ‘not-agreed-upon’ variants of core concept that create among those who try to use the model.

The Relational Data Model

The relational data model, first proposed in early 1970s by Dr E.F.Codd of IBM but not fully developed until the 1980s, is based on the mathematical theory of sets of relations. In this model, entities and relationship are presented in tables. Each horizontal row of each table describes a record, and each column describes one of the attributes (data fields) of the record. The simple structure of relational model has widespread appeal and has drawn supporters from computer users unhappy with the hierarchical and network data models. With this model, data can be entered into a database without too much thought about how it will be used. To add new data is like adding a new column of attributes or a new row at bottom of table. Programs can be written to manipulate or extract data from data base by using relational algebraic operations such as AND, OR, NOT, JOIN and PROJECT

However, to find a particular item, a search of all the table may have to be made by a computer system. Clearly this search process will be slow when database is large. Very fast hardware may reduce search time but raises the cost of processing. An index can be prepared of common search paths with index pointers to frequently sought of any item. So, to scan index is the first step of any search. It lengthens search time for non-indexed items and an index uses up computer storage space.

Which model is best?

Data models can be evaluated on the basis of usability, implementability and performance. The answer that DBMS data model is best depends on philosophical orientation. The hierarchical data model is the oldest model and the most popular on mainframes. People who think the users (or programmers) should be able to have some control over the details of storage allocation and search paths often prefer network systems. People who believe that users should be isolated from the mechanisms used to access data, and those who assign high priority to a model that is easy to understand and construct, frequently favor relational model.

Physical Data Base Design

Once the logical organization of data element is designed, analyst plans how to place logical data records and files on physical medium for storage until they are needed for processing. This called the physical design of the database. They try to plan storage so that data can be retrieved and updated as efficiently as possible given resource constraints.

Physical Design of File Systems

Physical design begins with the physical organization of data. This means organizing logical records and logical files into physical records and physical files.

The computer needs an aid to locate data belonging to a logical file in one or more physical files. Data is located by an address that 'point' to the data'location in a physical file. The space allocated a data element in a physical record is called a field.

In the early days of computing, data were stored on punched cards and the physical storage space was measured in columns. The analyst decided on the number of column for each data element, called width of field.

Nowadays, most logical records are stored on tape or disk. A header label is used at the beginning of each tape to identify the logical records on the tape. The logical records

themselves are separated by an inter-record gap, which identifies the end of one logical record and signals the start of another. In planning the location of data on tape, analysts prepare a data layout form which identifies each data element on the tape, its width of field and its sequencing. Each data element is located in processing by searching the entire tape.

The use of magnetic tape is appropriate for sequential processing when all logical records in a logical file need processing. But tape is inappropriate for handling individual logical records. When data of a single logical record is frequently needed, then a computer disk, a random storage device, is chosen by analyst. The reason is that data recorded on disk tracks can be accessed by positioning the read mechanism (or the disk itself) directly at the data required without having to search through the whole disk. This called direct access or random access

Physical Design of a Data Base System

Most organizations that implement a database system depend on a DBMS for management of that system. So, systems analysts on development teams do not have to plan the physical design of the database. However, they will help decide which DBMS to acquire and subsequently implement the system.

3.7.2 An ideal database structure

Main components of a database consists of:

1 The Conceptual schema

The Conceptual schema is used to describe the application area, including a list of entity types and ties applied for relationship defined in the system design and analysis phase, oriented to users and system designers. A part of a conceptual schema concerned by one user or a group of users is called small conceptual schema. It is oriented more to users than to database designer.

2 The Database schema

The Database schema describes the data stored in a database and defines the different directions of access to the database. It also points out specifications of privacy and overall connection. However, database schema shows not only the way to store a database but also how access is provided. A part of database schema concerned by a user or a group of users is called “Small Database Schema”. This schema has various responsibilities

Assist programmers to seek for access paths in the database they are concerned to

Assist users to make up questions by Q&A manner

Easy to split database into units to depict particular limit (for instance, a small database schema may give all users access to data, but it define only one user as authorized to update the data)

3 The Physical schema

The Physical schema depicts the physical structure of a database. Building up the physical scheme is a main step in system design

4 The Physical storage structure

This is a structure which stores the database, including storage in diskettes, tapes, for the program to process

5 The Back-up and Recovery System

The Back up and Recovery System is a module which permits the recovery of a database after an unexpected accident in software or hardware.

6 The User interface

Software of user interface is mainly the application program and Q&A language. Users use application programs to enter, search and update database and replicate the data in report form. Background knowledge of database is not really required. The users must be informed of the common relationship of data and the meaning of notices given by the application program. However, in order to take advantage of the utilities supporting report generation such as Report Program Generator, users are required to be knowledgeable of the database contents (so that they can describe the data contents and report format). With regard to Q&A language, users also need to understand the data accessing paths defined in database. This information is obtained by referring to the corresponding small database schema. Similarly, application programmer also has to refer to small database schema when producing application program.

7 Mapping module from logical to physical

The transformation of logical database schema to physical storage structure is made automatically because the mapping module from logical model to physical model on the basis of information about the physical structure of database stored in physical schema.

8 Privacy sub-systems: Privacy subsystems protect the database from illegal access. The best way to display the private ties is by a common language with the language used in application programs, making the private ties be linked closely to database schema. When users take a function of the application program (data entry, report replication), the private

sub-system will base on database schema to check whether the accesses are legal and take relevant actions

9 Integrity sub-system: Integrity sub-system protects databases from being added with inaccurate data types. The common ties are presented by the common language used in application programs and thus, they are linked tightly to database schema. When users take the function of data entry of the application program, Integrity sub-system will base on database schema and the database itself to check whether the access is permitted.

3.7.3 Physical database design

The objective of this phase is to give the system the definitions of the data and to build the data structure (file/table). The installation relies on the following information:

- Ties of user's system implementation
- Details of data usage analysis: data model, relationship model, path analysis diagram, navigation model and data usage chart.

Designers often use "First Cut Rule" in logical data model to transform it into the initial gathering of files and tables. The "First Cut Rule" is the rule that carry out these tasks: Remove the redundant relationships; Determine all the relationship types and convert them to 1-N relationship type. They then process to fine-tune these files until they meet the effectiveness requirement of the system. This fine tuning process starts from the adjustment of data usage model or at least of the most important files. Only in this way the system's ties and requirements will be met.

Before you create a First – cut Data Design, take time to plan what exactly is needed. You will need a fully documented information model, and some time to think.

Most of a First – cut Data Design can be derived from an information model by performing some basic mappings.

- Entities become tables: the entity name is, by convention, made plural as the resulting table contains a set of entity occurrences.
- Attributes become columns: the characteristic of the column is defined from that of the attribute, for example, mandatory attribute: NOT NULL column.
- Unique identifiers become primary and unique key constraints.
- Relationships become foreign key columns and constraints.

However, in practice, your information model might not be that simple, if it contains:

- An entity with several unique identifiers; choose the most appropriate candidate to become the primary key constraint. The remaining identifier(s) become unique key constraints.
- An entity without a unique identifier; create a primary key column and constraint for the corresponding table. This is known as a surrogate or artificial key.
- Sub – type entities, or an arc structure; you need to decide how they are to be implemented in your data design.
- A many to many relationship, either: Revisit your analysis and resolve each many to many relationship into two one to many relationships with an intersection entity or leave it as a many to many relationships and allow the Database Design Wizard to create intersection tables.

Note: Producing the first – cut is only the first stage in building a complete data design. You can compare it with laying the foundations of a building. Constructing a complete, well tuned and implemented database are topics for further discussion later.

3.7.4 Designing process

The transformation process from logical model into physical database is carried out in 2 phases:

1 Apply several standard rule – “First Cut Rules” whereby the designer can easily create a physical database structure which is at least workable though still ineffective. Each database management system has its own “First Cut Rules”. As for relationship database (Oracle, Dbase) the transformation requires a process. An entity type in the model will be mapped to a separate table with the assistance of various tools. Moreover, while analyzing data, we have created a relationship model consistent to the structure of the relationship database management system. That’s why we can easily install this relationship model.

2 Optimization: After creating the first cut, database designers will modify and improve the database structure base on the data diagram by taking repetition in the database, recommend various accessing methods, integrate entity types to get the answer more quickly. To do this, designers need to have expertise in database management system. Although optimization is not a subject of this material, the testing process should be made in a physical file, then check whether the structure proposal is for all database management systems.

3.7.5 Physical storage structure design

The way designers decide to store the data physically to secure the fastest accessing, processing and data replicating is affected by different elements: data specification described in physical schema, user interface, hardware features, operating system, programming language, staff experience, special requirements of application's management.

Designers should note the following important aspects while designing the physical storage structure:

Data independence: This term is used to depict the extent to which the physical storage structure is independent with the application program accessing to it and vice versa. An independent data system has the following features:

- a. All relationships between entities described in conceptual schema are demonstrated by effective accessing paths in the physical storage structure
- b. All accessing requirements by application programs sent to physical schema are presented by a logical language independent with the data structure being used, including physical storage structure. Such requirements are transformed from logical to physical by mapping the modules into dependent implementation commands. For instance, the logical requirement "looking for the book Structured Analysis and Design" needs to be transformed to the following command: look for entries in file BOOK, using book title as key word and produce a list of writers of Structured System Analysis and Design.

Advantages of an independent data database are:

- a. New application programs can be added to the system at minimum costs. If the required accessing path already exists, it is unnecessary to modify or enlarge the physical storage structure

The installation environment can be changed more easily. When there is a new computer, which requires a new data structure is used in the physical storage structure, the changes can be made easily without any alteration to application program. The only program that is changed is the module mapped from logical to physical.

In fact, physical storage structures are designed with flexibility for the system to be developed.

Privacy

Database privacy is defined as an extent to which data are protected from illegal access.

This privacy requirement is defined before the analysis process. One of the privacy requirement of the library book management system is: librarian is the only person authorized to update book-returning ticket (delete names of borrowers of the book). In the process of designing database schema, such requirements will be transformed to access ties demonstrated by data factors and accessing paths. Thus, the above requirements can be transformed to the following tie: Librarian is the only user authorized to go along the intersection access path to borrowing ticket

Though the accessing ties are considered in the designing process of the physical storage structure, the best way to control the accessing ties is to put them in a private module or sub-system. This approach has 2 main advantages:

- ã Facilitating the designing process. All access controlling mechanism are gathered in one module and can be installed by the standard approach
- ã The systems obtained are more data independent if the access controlling mechanism is inserted into the physical storage structure or application programs.

This private module can be a simple program with the following functions:

- ã Identify users
- ã List components of database schema corresponding to the components of database that users wish to have access to
- ã Check accessing ties to see whether an access is legal. If it is, the access is permitted, and if it isn't, take the actions as defined in the database schema.

When a subsystem is exclusively designed, it will result in a number of decisions relating to users. With the privacy requirements defined in the designing process, users are authorized to have access to data by using their Ids, passwords or magnetic cards accepted by the computers

In fact, application programs tend to generate accessing requirements more in the physical form than in the logical data structure. In most cases, the privacy requirements must be described in detail. Listed below are the most popular kinds of tie:

- ã Privacy ties depicted as users' accessing authority exclusively to the parts of physical storage structure, such as limiting access to records of data table

- ~b Some ties limit the access to several functions of the program or relevant data files. This way of controlling can be done by using password to limit or forbid access to some functions and files
- ~c Other ties may require special procedure and are often located at in/out application program. The procedures are used to limit access to some parts of database which is not considered by operating system as data unit. For example, a user is only permitted to read the data in some fields of a certain record. This kind of control asks for a relevant procedure.

Integrity sub-system designing

Integrity sub-system is responsible for maintaining the integrity by protecting database against illegal changes and damages in the whole system. Data errors can be seen in the following cases:

- ~a Data is not updated frequently. For example, a book coded xxx that has been lost for some reasons is still listed in the book list of the library
- ~b Data is updated simultaneously
- ~c Data is damaged while being coded, changed or transmitted from a distance
- ~d Data is damaged while being stored due to hardware faults.

It is important to take the right technique to minimize the above errors or to identify them once they exist

It is necessary to generate data testing protocol (according to the ties required) and insert it to application program to test input data. It is hard to maintain a flawless system, however, the database system should be designed in a way that it can best maintain the security and accuracy of the data. One of the most important aspects to secure the database against errors is to back-up the data regularly and recover the data when necessary

- ~a Lock strategy: The database should be classified into various units to be updated. The units can be field, record, file or a larger part of the database. While a replica of the unit type chosen (field, record...) is updated, the original must be locked and no access to it is permitted. When the update is finished, the new version of the unit will supersede the old version. In this way, if the system encounters accident while updated, the original version is still kept.

↳ **Data back up:** Back up files include diary and stored files. Normally, diary is a file stored in a tape containing replica (or features) of database units before or after they are updated. Stored files consist of the replica of the entire or parts of the database which are stored in a circle. For example, the replica of a part of the database can be made everyday while the copy of the entire database can be made once a week.

↳ **Recovery:** A way to recover a version of the database after the system encounters accident, meaning that the database must be able to go back to its status before the accident. The following techniques can be used in this case:

Roll back: The diary file is researched to define the previous image of the units updated when the system was down. Then the images will be recovered

Roll forward: A replica of the database (or the damaged part of the database) is recovered from the latest file and fed in the system. The diary file is used to take the database to the status before the accident.

Database design in the presented example:

Base on the entity diagrams in the system analysis phase, we continually design the tables correspondent with them as the following rules:

- Entity** becomes **Table**
- Attribute** becomes **Column (field)**
- unique identifier** becomes **primary key**

The field name, data type, width and value (input value or not) in the library management system are listed in the tables below:

1. LANGUAGE TABLE:

Field name	Data Type	Width	Value
LANG_NO	CHARACTER	5	NOT NULL
LANG_NAME	CHARACTER	40	NOT NULL
LANG_VN	CHARACTER	40	NOT NULL

LANG_SYS	CHARACTER	3	NOT NULL
----------	-----------	---	----------

2. NATION TABLE:

Field name	Data Type	Width	Value
NATION_NO	CHARACTER	5	NOT NULL
NATION_NAME	CHARACTER	40	NOT NULL
NATION_VN	CHARACTER	40	NOT NULL

3. COLLECTION TABLE:

Field name	Data Type	Width	Value
COLL_NO	CHARACTER	5	NOT NULL
COLL_NAME	CHARACTER	60	NOT NULL

4. SPECIALITY TABLE:

Field name	Data Type	Width	Value
SPEC_NO	CHARACTER	5	NOT NULL
SPEC_NAME	CHARACTER	50	NOT NULL

5. DEPARTMENT TABLE:

Field name	Data Type	Width	Value
DEPT_NO	CHARACTER	5	NOT NULL
DEPT_NAME	CHARACTER	60	NOT NULL

6. READER TABLE:

Field name	Data Type	Width	Value
READER_NO	CHARACTER	20	NOT NULL
READER_NAME	CHARACTER	40	NOT NULL
DEPT_NO	CHARACTER	5	NOT NULL
ADDRESS	CHARACTER	100	NULL
BIRTH_DATE	DATE		NULL
COMMENT	CHARACTER	200	NULL

7. BOOK TABLE:

Field name	Data Type	Width	Value
BOOK_NO	CHARACTER	20	NOT NULL
BOOK_NAME	CHARACTER	100	NOT NULL
SIZE	CHARACTER	50	NOT NULL
TIME_PUBL	NUMBER	3	NOT NULL
YEAR_PUBL	CHARACTER	4	NOT NULL
PUB_HOUSE	CHARACTER	100	NOT NULL
COST	NUMBER	7	NULL
AUTHOR	CHARACTER	50	NOT NULL
CHIEF_AUTH	CHARACTER	50	NULL
COMP_AUTH	CHARACTER	50	NULL
REV_AUTH	CHARACTER	50	NULL
LANG_NO	CHARACTER	5	NOT NULL
NATION_NO	CHARACTER	5	NOT NULL
SPEC_NO	CHARACTER	5	NOT NULL
COLL_NO	CHARACTER	4	NOT NULL
KW_MASTER	CHARACTER	40	NULL
KW_SLAVE	CHARACTER	40	NULL
COMMENT	CHARACTER	200	NULL

8. L_GB_BOOK_TICKET TABLE:

Field name	Data Type	Width	Value
READER_NO	CHARACTER	20	NOT NULL
BOOK_NO	CHARACTER	20	NOT NULL
LEND_DATE	DATE		NOT NULL
GIVEBACK_DATE	DATE		NULL
COMMENT	CHARACTER	200	NULL

9. MAGAZINE_HEADER TABLE:

Field name	Data Type	Width	Value
MAG_HEAD_NO	CHARACTER	20	NOT NULL
MAG_NAME	CHARACTER	100	NOT NULL
START_YEAR	CHARACTER	4	NOT NULL
MAG_SHELF	CHARACTER	100	NULL
ISSN_NO	CHARACTER	30	NULL
PUB_HOUSE	CHARACTER	100	NULL
LANG_NO	CHARACTER	5	NOT NULL
NATION_NO	CHARACTER	5	NOT NULL
SPEC_NO	CHARACTER	5	NOT NULL
COLL_NO	CHARACTER	5	NOT NULL
COMMENT	CHARACTER	200	NOT NULL

10. MAGAZINE_DETAIL TABLE:

Field name	Data Type	Width	Value
MAG_HEAD_NO	CHARACTER	20	NOT NULL
MAG_DETL_NO	CHARACTER	20	NOT NULL
YEAR	CHARACTER	4	NOT NULL
VOLUME	NUMBER	4	NULL
NUMBER	NUMBER	4	NULL
MONTH	CHARACTER	2	NULL
QUANTITY	NUMBER	4	NOT NULL

Questions

- 1̃ What is Structured Design?
- 2̃ How do system analysis and design relate to each other?
- 3̃ What are the main phases of the design process?
- 4̃ What are the main tools and design techniques?
- 5̃ What is logical design and its main tasks?
- 6̃ What is physical design and its main tasks?
- 7̃ How do you distinguish between logical design and physical design?
- 8̃ Why do you need to identify the borderline of the computer system in the system DFD?
How do you present them in the model
- 9̃ What are the main components of computer-user interface design?
- 10̃ Why is it necessary to design system monitoring?
- 11̃ How do you organize the system's components?
- 12̃ What are the main modelling tool and techniques?
- 13̃ What are the tools used in data usage analysis?
- 14̃ What is navigation model and its purpose?
- 15̃ What are the main components of a database and the role of each component?

References

- 1 Bernard Boar, *Application Prototyping*. Reading, Mass.: Addison – Wesley, 1984.
- 2 Brian Dickinson, *Developing Structured Systems*. New York: Yourdon Press, 1981.
- 3 C.J. date, *An Introduction to Database Systems*, 4th ed. Reading, Mass.: Addison-Wesley, 1986.
- 4 Chris Gane and Trish Sarson, *Structured Systems Analysis and Design*. New York: Improved Systems Technologies, Inc., 1977.
- 5 D.C. Tsichritzis and F.H. Lochovsky, *Data Models*. Englewood Cliffs, N.J.: Prentice-Hall, 1982.
- 6 Edward Yourdon, *Managing the Systems Life Cycle*, 2 nd ed. Englewood Cliffs, N.J.: Prentice-Hall, 1988.
- 7 Edward Yourdon and Larry L. Constantine, *Structured Design: Fundamentals and Applications in Software Engineering*, 2 nd ed. Englewood Cliffs, N.J.: Yourdon Press, 1989.
- 8 J.D. Lomax, *Data Dictionary Systems*. Rochell Park, N.J.: NCC Publications, 1977.
- 9 Shaku Atre, *Data Base: Structured Techniques for Design, Performance, and management*. New York: Wiley, 1980.
- 10 K.M. Hussain and Donna Hussain, *Information Systems: Analysis, Design and Implementation*. Tata McGraw-Hill, 1995.
- 11 K.T. Orr, *Structured System Development*. New York: Yourdon Press, 1977.
- 12 Lawrence peters, *Advanced Structured Analysis and Design*. Prentice-Hall, 1987.
- 13 Lavette C. Teague, Jr., and Christopher Pidgeon, *Structured Analysis Methods for Computer Information Systems*. Chicago: Science Research Associates, 1985.
- 14 Marjorie Lesson, *System Analysis and Design*. Chicago: Science Research Associates, 1981.
- 15 Meilir Page-Jones: *The Practical Guide to Structured Systems Design*, 2 nd ed. Englewood Cliffs, N.J.: Yourdon Press, 1988.
- 16 Peter Chen, *The Entity-Relationship Approach to Logical database Design*. Wellesley, Mass.: Q.E.D. Information Sciences, 1977.

17 Tom DeMarco, *Structured Analysis and System Specification*. New York: Yourdon Press, 1978.

18 *Application design using Oracle designer/2000*

19 *Relation System Design* (Oracle)

20 *Detailed Systems Analysis* (Oracle)

21 *Introduction Approaches to data Design* (Oracle)

22 *Model Business Systems with Designer/2000* (Oracle)

INDEX

A

Attributes: 3.1.1, 3.7.1, 3.7.3

C

Cohesion: 3.1.2, 3.1.6, 3.2.1

Computer system: 3.1.5, 3.1.6, 3.2.1, 3.2.2, 3.5.3, 3.6, 3.6.3, 3.7.1

Coupling: 3.1.2, 3.1.6, 3.2.1

D

Data dictionary: 2.1.1, 2.1.6, 2.2, 2.4.2, 2.4.3, 3.1.3, 3.1.6, 3.2.2

Data flow diagram (DFD): 1.2.5, 2.1.1, 2.1.6, 2.2, 2.4.1 – 2.4.3, 3.1.6, 3.2.1

Data model: 3.1.6, 3.2.1, 3.6, 3.6.1, 3.6.2, 3.7.1, 3.7.3

Data usage: 3.6, 3.6.2, 3.6.3, 3.7.3

Database design: 3.1.3, 3.1.6, 3.7.1, 3.7.3, 3.7.5

Database structure: 3.5.3, 3.7.2, 3.7.4

Definition: 1.1.1, 2.1.2, 2.1.6, 2.2.4, 2.3.2, 3.1, 3.1.6, 3.7.4

Definition of: 1.1.1, 2.2.4

Detailed Design: 1.1.3, 3.2.1

E

Entity-Relationship diagram (ERD): 2.1.1, 2.1.3, 2.1.6, 2.2, 2.4.2, 2.4.3, 3.1, 3.2.1

Entity Relationship model: 2.2.2, 3.1.6

Example: 1.1.1, 1.2.3, 1.2.4, 2.1.6, 2.2.2, 2.2.4, 2.3.1, 2.3.2, 3.2.1, 3.2.2, 3.3.3,
3.6.1, 3.7.1, 3.7.3, 3.7.5

F

First - Cut: 3.2.1, 3.7.3, 3.7.4

Function analysis: 1.2.5, 2.1.6

Function diagram: 1.2.5, 2.1.1, 2.1.6, 2.2, 2.4.2, 2.4.3

H

Hierarchy: 2.1.6, 2.4.3, 3.1.2, 3.1.6, 3.2.1, 3.3.3, 3.5.2

I

Intersection: 3.5.3, 3.7.3

L

Library management: 1.2.3, 1.2.4, 1.2.5, 2.1.6, 2.3.1, 2.3.2, 3.2.1, 3.3.3, 3.7.5

Library manages: 1.2.4, 2.3.1

Life cycle: 1.1.3, 2.1.1, 2.1.3, 2.4.2, 3.1.6

Logical design: 3.1.1, 3.2.1

Logical model: 2.1.1, 3.2.1, 3.2.2, 3.7.2, 3.7.4

M

Management: 1.1.2, 1.2.1, 1.2.2 – 1.2.5, 2.1.4, 2.1.6, 2.3.1, 2.3.2, 2.4.1, 2.4.3,
3.1, 3.1.6, 3.2.1, 3.2.2, 3.3.3, 3.3.4, 3.7.1, 3.7.4, 3.7.5

Method: 1.1.2, 1.2, 1.2.3, 1.2.4, 2.1, 2.1.1, 2.1.2, 2.1.5, 2.1.6, 2.2, 2.2.1, 2.2.2,
2.2.4, 2.3, 2.4.1 – 2.4.3, 3.1.1, 3.1.6, 3.4, 3.7.1, 3.7.4

Methodology: 2.1.1, 2.2, 2.3, 3.1.6

N

Navigation model: 3.6, 3.6.2, 3.7.3

Normalization: 2.3, 2.3.1, 2.3.2, 2.3.4

P

Phase of: 3.2.1, 3.5, 3.5.2, 3.7.1

Physical Design: 3.1.1, 3.1.6, 3.2.1, 3.7.1

Physical model: 2.1.1, 3.7.2

Primary key: 3.7.3, 3.7.5

Process specification: 2.1.1, 2.1.6, 2.2

Pseudocode: 3.1.1, 3.1.3, 3.1.6, 3.2.1

R

Relationship: 3.1, 3.1.1, 3.1.3, 3.1.6, 3.5.1, 3.5.3, 3.6.1, 3.6.2, 3.7.1 – 3.7.5

Requirement: 1.1.2, 1.1.3, 1.2.1 – 1.2.5, 2.1.1, 2.1.4 – 2.1.6, 2.2.1, 2.2.2, 2.3,
2.4, 2.4.1 – 2.4.3, 3.1.2, 3.1.5, 3.1.6, 3.2.1 – 3.2.3, 3.6, 3.6.3, 3.7.1, 3.7.3, 3.7.5

S

Spiral model: 1.2, 2.1.1

Stage of: 1.1.1, 2.3.1

Store: 1.2.3, 1.2.4, 2.1.6, 2.2.2 – 2.2.4, 2.3.1, 3.1.5, 3.1.6, 3.7.1, 3.7.2, 3.7.5

Structured design: 3.1.1 – 3.1.3, 3.1.6

Structured system: 2.1, 2.1.1, 2.1.6, 3.7.5

Structured system analysis: 2.1.1, 2.1.6, 3.7.5

Survey: 1.2.2 – 1.2.5, 2.1, 2.1.1, 2.1.3, 2.2.3, 2.3.2 – 2.3.4, 2.4.1

System analyst: 1.1.2, 2.1.3, 2.1.4, 2.1.6, 2.2.1, 2.2.2, 2.4.1

System analysis: 1.2.3, 2.1, 2.1.1 – 2.1.3, 2.1.5, 2.1.6, 2.2.4, 2.3.2, 2.4, 2.4.1 –
2.4.3, 3.1.3, 3.2.1 – 3.2.3, 3.7.5

System design: 1.1.2, 2.1.2, 2.1.3, 2.1.6, 2.4.3, 3.1.5, 3.1.6, 3.2.1, 3.2.2, 3.5.2,
3.7.2

System designers: 1.1.2, 2.1.2, 2.4.3, 3.1.5, 3.2.1, 3.2.3, 3.7.2

System development: 1.1.2, 1.1.3, 1.2, 1.2.1, 1.2.4, 2.1.1, 2.1.2, 2.4.2, 3.1.6, 3.4,
3.6.2

T

Techniques: 1.1.1 – 1.1.3, 2.1.1, 2.1.5, 2.2, 2.2.4, 2.3.1, 2.3.4, 2.4, 2.4.3, 3.1.6,
3.2.1, 3.5.3, 3.7.1, 3.7.5

Techniques of: 1.1.1, 3.1.6

Tools: 1.1.2, 1.2.1, 1.2.2, 2.1, 2.1.1, 2.1.3, 2.1.5, 2.1.6, 2.2, 2.2.2, 2.2.4, 2.4,
2.4.1 – 2.4.3, 3.1.6, 3.2.1, 3.5.3, 3.6, 3.6.2, 3.7.1, 3.7.4

U

Users: 1.1.2, 1.1.3, 1.2, 1.2.1 – 1.2.5, 2.1.1 – 2.1.6, 2.2.1, 2.2.2, 2.3.1, 2.3.2,
2.3.4, 2.4, 2.4.1 – 2.4.3, 3.1.6, 3.2.2, 3.3.1 – 3.3.3, 3.4, 3.5.1 – 3.5.3,
3.7.1, 3.7.2, 3.7.5

W

Waterfall model: 1.2, 2.1.1