PERIYAR INSTITUTE OF DISTANCE EDUCATION (PRIDE)

PERIYAR UNIVERSITY

SALEM - 636 011.

B.Sc. COMPUTER SCIENCE

SECOND YEAR

PAPER - III : SYSTEM ANALYSIS AND DESIGN

Prepared by :

Prof N. RAJENDRAN, M.C.A.,M.Phil.,

HOD of Computer Science,

Vivekanandha College of Arts and Sciences for Women,

Elayampalayam, Tiruchengode,

Namakkal (District) – 637 205.

B.Sc. COMPUTER SCIENCE
SECOND YEAR

PAPER - III : SYSTEM ANALYSIS AND DESIGN

FOREWORD

"In human affairs we have reached a point where the problems that we must solve are no longer solvable without the aid of computers. I fear not computers but the lack of them"

- Issac Asimov

Dear Students,

We are in the block of course **System Analysis and Design**. In this book we will discuss about analysis of systems, systems development life cycle, files organization and software Engineering concepts.

In the first unit you are going to learn about the concepts of information systems and designs.

In the second unit the details of the tools for determining system requirements are discussed.

In the third unit the analysis of design transitions has been explored elaborately.

In the fourth unit a clear idea of design of online dialogue and design of files and methods of file organization is given.

In the fifth unit we can gain knowledge on the areas of system Engineering and quality assurance.

PRIDE would be happy in you could make use of this learning material to enrich your knowledge and skills to serve the society.

B.Sc. COMPUTER SCIENCE
SECOND YEAR

PAPER - III : SYSTEM ANALYSIS AND DESIGN

BLOCK

SYSTEM ANALYSIS AND DESIGN

UNIT-I

Introduction to Information System Development-What is System Analysis and design? -Business system concepts-Categories of Information systems-System development strategies. Managing the application development portfolio: How system projects are begun? - Managing project review and selection-Preliminary investigation-Selecting the project development strategies.

UNIT-II

Tools for determining system requirement: What requirements determination?-Fact finding technique-Tools for documenting procedure and decision. Structure analysis development strategies: Structured Analysis Developing Data flow diagram. Computer Aided system tools: Role of tools-Categories of automated tools-CASE Tools-Benefits of CASE.

UNIT-III

The Analysis to design transactions: Specifying application requirements-Objectives in designing information system-What features must be designed? Design of computer output: How to identify computer output needs- how to present information-Designing printed output: What concerns guide input design-Structuring data for input-Input validation.

UNIT-IV

Design of online dialogue: How is online different?-What is an interface-Designing dialogue-Dialogue strategy data entry dialogues. Design of files and use of auxiliary storage devices: Basic file terminology-Data structure programs-Types of files-Methods of file organization.

UNIT-V

Systems engineering and quality assurance: Design objectives-Program structure charts-Design of software-Managing Quality assurance-Managing testing practices. Managing system implementation: Training-Conversion-implementation review. Managing information systems development: Estimation and management of development Estimation-Personnel and development management. Hardware and software selection: Hardware selection-Software Selection.

SYSTEM ANALYSIS AND DESIGN

UNIT-I

- Introduction to Information System Development.
- What is System Analysis and design?
- Business system concept.
- Categories of Information systems
- System development strategies.
- Managing the application development portfolio.
- How system projects are begun?
- Managing project review and selection.
- Preliminary investigation.
- Selecting the project development strategies.

UNIT-II

- Tools for determining system requirement.
- What requirements determination?
- Fact finding techniques.
- Tools for documenting procedure and decision.
- Structure analysis development strategies.
- Structured Analysis.
- Developing Data flow diagram.
- Computer Aided system tools: Role of tools-Categories of automated tools-CASE Tools-Benefits of CASE.

UNIT-III

- The Analysis to design transactions.
- Specifying application requirements.
- Objectives in designing information system.
- What features must be designed?
- Design of computer output.
- How to identify computer output needs?
- How to present information?
- Designing printed output.

- What concerns guide input design?
- Structuring data for input.
- Input validation.

UNIT-IV

- Design of online dialogue.
- How is online different?
- What is an interface?
- Designing dialogue.
- Dialogue strategy data entry dialogues.
- Design of files and use of auxiliary storage devices.
- Basic file terminology.
- Data structure programs.
- Types of files.
- Methods of file organization.

UNIT-V

- Systems engineering and quality assurance.
- Design objectives.
- Program structure charts.
- Design of software.
- Managing Quality assurance.
- Managing testing practices.
- Managing system implementation.
- Training & Conversion.
- Implementation review.
- Managing information systems development.
- Estimation and management of development Estimation.
- Personnel and development management.
- Hardware and software selection.

UNIT –I : Introduction to Information System Development

UNIT STRUCTURE

UNIT –I : Introduction to Information System Development

1.0 Information System

Definition

An information system is an open system that allows inputs and facilitates interaction with the user .The main characteristics of an open system are input from outside, processing, output, operating in cycles through feedback, differentiation and equifinality.

There are three levels of information in organizations that require a special type of information system:

a) ***Strategic*** information relates to long-range planning policies and upper management. It is achieved with the aid of decision support systems.

b) ***Managerial*** information helps middle management and department heads in policy implementation and control. It is maintained with the aid of management information systems.

c) ***Operational*** information is daily information needed to operate the business. It is established by data processing systems.

The nature of information and managerial levels is related to whether decision-making is structured or unstructured. A relatively closed, stable and mechanistic business tends to develop a more structured information process for planning and control. Open, dynamic organizations are associated with uncertain environment where more unstructured rather than structure decisions are made to cope with uncertainty.

1.1 INTRODUCTION

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design, mainly deals with the software development activities.

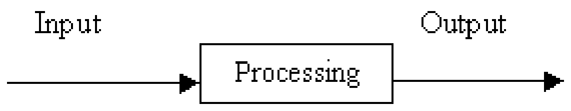
1.2 OBJECTIVES

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

- Understand a system
- Understand the different phases of system developments life cycle
- Know the components of system analysis
- Know the components of system designing

1.3 Defining A System

A collection of components that work together to realize some objective forms a system. Basically there are three major components in every system, namely input, processing and output.



In a system the different components are connected with each other and they are interdependent. For example, Human body represents a complete natural system. We are also bound by many national systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs.

1.4 SYSTEM LIFE CYCLE

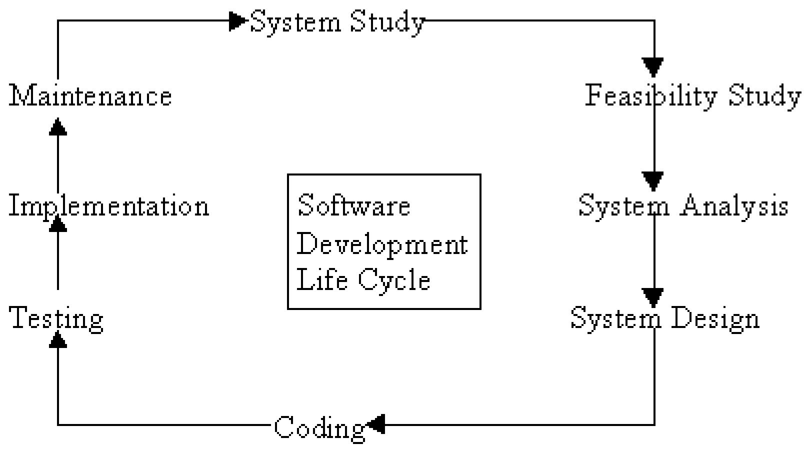
System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required developing a system.

System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle means software development life cycle.

Following are the different phases of software development cycle:

- System study
- Feasibility study
- System analysis
- System design
- Coding
- Testing
- Implementation
- Maintenance

The different phases of software development life cycle is shown in the following figure



Different phases of Software development Life Cycle

1.5 PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

Let us now describe the different phases and the related activities of system development life cycle in detail.

(a) System Study

System study is the first stage of system development life cycle. This gives a clear picture of what actually the physical system is? In practice, the system study is done in two phases. In the first phase, the preliminary survey of the system is done which helps in identifying the scope of the system. The second phase of the system study is more detailed and in-depth study in which the identification of user's requirement and the limitations and problems of the present system are studied. After completing the system study, a system proposal is prepared by the System Analyst (who studies the system) and placed before the user. The proposed system contains the findings of the present system and recommendations to overcome the limitations and problems of the present system in the light of the user's requirements.

To describe the system study phase more analytically, we would say that system study phase passes through the following steps:

- Problem identification and project initiation
- Background analysis
- Inference or findings

(b) Feasibility Study

On the basis of result of the initial study, feasibility study takes place. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness. The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy.

(c) System Analysis

Assuming that a new system is to be developed, the next phase is system **analysis**. Analysis involved a detailed study of the current system, leading to specifications of a new system. Analysis is a detailed study of various operations performed by a system and their relationships within and outside the system. During analysis, data are collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are the tools used for system analysis. Using the following steps it becomes easy to draw the exact boundary of the new system under consideration:

- Keeping in view the problems and new requirements
- Workout the pros and cons including new areas of the system

All procedures, requirements must be analyzed and documented in the form of detailed data flow diagrams (DFDs), data dictionary, logical data structures and miniature specifications. System Analysis also includes sub-dividing of complex process involving the entire system, identification of data store and manual processes.

The main points to be discussed in system analysis are:

- Specification of what the new system is to accomplish based on the user requirements.
- Functional hierarchy showing the functions to be performed by the new system and their relationship with each other.
- Function network, which are similar to function hierarchy but they highlight those functions, which are common to more than one procedure.
- List of attributes of the entities - these are the data items which need to be held about each entity (record)

(d) System Design

Based on the user requirements and the detailed analysis of a new system, the new system must be designed. This is the phase of **system designing**. It is a most crucial phase in the development of a system. Normally, the design proceeds in two stages:

- Preliminary or general design
- Structure or detailed design

Preliminary or general design: In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to the detailed design stage.

Structure or Detailed design: In the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured. Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationship among the same components as the original problem. Input, output and processing specifications are drawn up in detail. In the design stage, the programming language and the platform in which the new system will run are also decided.

There are several tools and techniques used for designing. These tools and techniques are:

- Flowchart
- Data flow diagram (DFDs)
- Data dictionary
- Structured English
- Decision table
- Decision tree

Each of the above tools for designing will be discussed in detailed in the next lesson.

(e) Coding

After designing the new system, the whole system is required to be converted into computer understanding language. **Coding** the new system into computer programming language does this. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer as **programs**. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future change, if required.

(f) Testing

Before actually implementing the new system into operations, a test run of the system is done removing all the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results.

Using the test data following test run are carried out:

- Unit test
- System test

Unit test: When the programs have been coded and compiled and brought to working conditions, they must be individually tested with the prepared test data. Any undesirable happening must be noted and debugged (error corrections).

System Test: After carrying out the unit test for each of the programs of the system and when errors are removed, then system test is done. At this stage the test is done on actual data. The complete system is executed on the actual data. At each stage of the execution, the results or output of the system is analyzed. During the result analysis, it may be found that the outputs are not matching the expected out of the system. In such case, the errors in the particular programs are identified and are fixed and further tested for the expected output. When it is ensured that the system is running error-free, the users are called with their own actual data so that the system could be shown running as per their requirements.

(g) Implementation

After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice. During this phase, all the programs of the system are loaded onto the user's computer. After loading the system, training of the users starts. Main topics of such type of training are:

- How to execute the package
- How to enter the data
- How to process the data (processing details)
- How to take out the reports

After the users are trained about the computerized system, manual working has to shift from manual to computerized working. The following two strategies are followed for running the system:

i. **Parallel run:** In such run for a certain defined period, both the systems i.e. computerized and manual are executed in parallel. This strategy is helpful because of the following:
    - Manual results can be compared with the results of the computerized system.

- Failure of the computerized system at the early stage, does not affect the working of the organization, because the manual system continues to work, as it used to do.

i. **Pilot run:** In this type of run, the new system is installed in parts. Some part of the new system is installed first and executed successfully for considerable time period. When the results are found satisfactory then only other parts are implemented. This strategy builds the confidence and the errors are traced easily.

(h) Maintenance

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment. It has been seen that there are always some errors found in the system that must be noted and corrected. It also means the review of the system from time to time. The review of the system is done for:

- Knowing the full capabilities of the system
- Knowing the required changes or the additional requirements
- Studying the performance

If a major change to a system is needed, a new project may have to be set up to carry out the change. The new project will then proceed through all the above life cycle phases.

1.6 Categories of Information Systems

Five Types of Information Systems

Information systems are constantly changing and evolving as technology continues to grow. Very importantly the information systems described below are not mutually exclusive and some (especially Expert Systems, Management Information Systems and Executive Information Systems are can be seen as a subset of Decision Support Systems). However these examples are not the only overlaps and the divisions of these information systems will change over time.

At present there are five main types:

Transaction Processing Systems (TPS)

Decision Support Systems (DSS)

Expert Information Systems (EIS)

Management Information Systems (MIS

Office Automation Systems (OAS)

The characteristics of an information system are:

The Organization of data into information

Ability to Analyse the Information

Processing Transactions

- Provide Users with Information About an Organization
- Help Decision-making
- Manage Information used Within an Organization

1.7 Examples of Database Information Systems

Most Information systems store data in a database and are referred to as Database management Systems. Examples referred to in the syllabus include:

1. A school database holding information on teachers, subjects, classrooms and students
2. The Roads and Traffic authority holding information on automobiles and holders of driver's licenses.
3. Video stores holding information on borrowers and videos.

1.8 Systems Development Strategies

- Adam Smith's book, *"Wealth of Nations"* (1776), was influential for the Industrial Age. It described the evolution from the Agricultural Age to the Industrial Age. It was the foundation for most industrial enterprises in the late 18th Century and into the 19th Century.
- By the middle of the 20th Century, the Industrial Enterprise had evolved into a complex series of manual processes. The pace of progress had seen most enterprises evolve to use increasingly complex business processes, with rapidly growing transaction volumes to be manually processed. These enterprises found … *they were operating in a continual state of Manual Chaos!*
- From the late 1950s, manual processes were automated by computer. We took the existing manual processes and then automated them essentially AS-IS: without much change. In so doing, we moved from Manual Chaos … *instead to Automated Chaos!*
- With rapid acceptance of the Internet in the second half of the 90s the chaos moved from the back office … onto the front doorstep of enterprises: through their web sites. We saw that customers could visit these enterprises by the click of a mouse. But they could just as quickly leave … *if the processes did not provide what the customers needed.*
- The problem is that we have 21st Century Enterprises that use 21st Century Technologies … *yet most enterprises today still use 18th Century Disintegrated Business Processes!*

- The business processes - originally designed based on principles set by Adam Smith in 1776 - have not evolved to take advantage of the technologies we have today. We need integrated 21st Century Enterprises together with integrated 21st Century Processes!
- We discussed the problem of redundant data versions in most enterprises. When data values change, all redundant versions must be updated to synchronize with that change. But with redundant data, *we moved to Data Maintenance Chaos!*
- We also discussed evolution of Systems Development Methodologies: Software Engineering; Information Engineering; and Object-Oriented methods.

1.9 ELECTING THE PROJECT DEVELOPMENT STRTEGIES

- We saw that in spite of these methods, process changes that require procedural program changes resulted in *Program Maintenance Chaos!* Our procedural programming methods were not designed so they could accommodate change easily, and object-oriented methods could not identify enterprise-wide reusable code.
- We saw that many Data Maintenance and Program Maintenance problems are resolved by *Business Integration*. We learned that Business Integration is best achieved by using Enterprise Architecture.
- We discussed Enterprise Architecture and the concepts of the *Zachman Framework for Enterprise Architecture*. We saw that the true architects of an enterprise are not in IT:
- Enterprise Architects are the senior managers who set the directions for the future, based on business plans, strategies and processes for that future and its technologies.
- The future will be based on business processes that use the technologies of today and tomorrow … with strategic directions set by senior management.
- From these strategic directions, business experts and IT experts can then work together in a design partnership to address these needs of the future.
- We discussed the concepts of Enterprise Engineering for rapid definition and delivery of Enterprise Architecture.
- When the Zachman Framework for Enterprise Architecture is used from the perspectives of the Planner and the Owner rows, with Enterprise Engineering applied enterprise-wide, the reusable activities and processes within an enterprise can be readily identified.

The key messages that I want to leave you with - for evolution to the 21st Century Enterprise - are therefore these:

- Rather than continue to build systems based on operational processes that reflect the needs of the past, by basing our designs for the future on the business plans that define that future we can build systems that can be implemented rapidly and changed easily.
- We must design for tomorrow based on business plans for the future. Our designs must draw on the knowledge of senior management and their business experts, reflected in the strategic business plans of the enterprise.
- We must use activities and processes that have been defined enterprise-wide by business experts applying these plans throughout the enterprise, to identify reusable activities and processes.
- From this enterprise-wide perspective, we can implement these reusable business activities and processes as business objects that can be implemented once, yet shared many times.
- Once implemented, these systems can respond to rapid business change environments as we have today. We will no longer be tied to inflexible stovepipe systems that cannot be changed easily.
- We can build for this future using Enterprise Architecture and Enterprise Engineering, together with Object-Oriented methods and technologies such as Web Services and Service-Oriented Architecture (SOA) Business Process Management (BPM) XML-based languages: to implement in weeks or days what previously took years or months.

SELF-ASSESSMENT QUESTIONS: I

ANSWER THE FOLLOWING QUESTIONS

1. What is a system?
   ----

2. What is a business system?
   ----

3. Programmer converts the program Specifications into computer instructions, which we refer as ________.

UNIT QUESTIONS

1. What is system analysis and design?
2. What is an information system?
3. Explain the various categories of the system.
4. Explain the various project development strategies.
5. Describe the system development strategies in detail.

RECOMMENDETION FOR FURTHER READINGS

1. System analysis and design, Elias M.Awad.
2. System analysis and design, Lee.

ANSWERS OF SELF-ASSESSMENT QUESTIONS

1. A system is the one that allows inputs and facilitates interaction with the user.
2. A business system deals with the techniques that generate the business and properties of a business environment.
3. Programs.

NOTES

UNIT II TOOLS FOR DETERMINIG SYSTEM REQUIREMENTS

UNIT STRUCTURE

UNIT II TOOLS FOR DETERMINIG SYSTEM REQUIREMENTS

2.1 REQUIREMENTS DETERMINATION

- Describe options for designing and conducting interviews and develop a plan for conducting an interview to determine system requirements
- Design, distribute, and analyze questionnaires to determine system requirements
- Explain advantages and pitfalls of observing workers and analyzing business documents to determine requirements

Learning Objectives

- Explain how computing can provide support for requirements determination
- Learn about Joint Application Design (JAD)
- Use prototyping during requirements determination
- Select the appropriate methods to elicit system requirements
- Apply requirements determination to Internet applications

2.2 FACT FINDING TECHNIQUES

- Gather information on what system should do from many sources
    - Users
    - Reports
    - Forms
    - Procedures

Performing Requirements Determination

- Characteristics for gathering requirements
    - Impertinence
        - Question everything
    - Impartiality
        - Find the best organizational solution
    - Relaxation of constraints
    - Attention to detail
    - Reframing
        - View the organization in new ways

2.3 Deliverables and Outcomes

- Types of deliverables:
    - Information collected from users

    - Existing documents and files
    - Computer-based information
    - Understanding of organizational components
        - Business objective
        - Information needs
        - Rules of data processing
        - Key events

Traditional Methods for Determining Requirements

- Interviewing and Listening
    - Gather facts, opinions and speculations
    - Observe body language and emotions
    - Guidelines
        - Plan
            - Checklist
            - Appointment
        - Be neutral
        - Listen
        - Seek a diverse view

2.4 Tools for Documenting procedure and Decision

- Interviewing
    - Interview Questions
        - Open-Ended
            - No pre-specified answers
        - Close-Ended
            - Respondent is asked to choose from a set of specified responses
    - Additional Guidelines
        - Do not phrase questions in ways that imply a wrong or right answer
        - Listen very carefully to what is being said
        - Type up notes within 48 hours
        - Do not set expectations about the new system

2.5 STRUCTURED ANALYSIS DEVELOPMENT STRATEGIES

- Administering Questionnaires

        - More cost-effective than interviews
        - Choosing respondents
            - Should be representative of all users
            - Types of samples
                - Convenient
                - Random sample
                - Purposeful sample
                - Stratified sample
- Questionnaires
    - Design
        - Mostly closed-ended questions
        - Can be administered over the phone or in person
    - Vs. Interviews
        - Interviews cost more but yield more information
        - Questionnaires are more cost-effective

Traditional Methods for Determining Requirements

- Interviewing Groups
    - Advantages
        - More effective use of time
        - Enables people to hear opinions of others and to agree or disagree
    - Disadvantages
        - Difficulty in scheduling
    - Nominal Group Technique
        - Facilitated process to support idea generation by groups
        - Individuals work alone to generate ideas which are pooled under guidance of a trained facilitator

Traditional Methods for Determining Requirements

- Directly Observing Users
    - Serves as a good method to supplement interviews
    - Often difficult to obtain unbiased data
        - People often work differently when being observed

2.6 Analyzing Procedures and Other Documents

- Types of information to be discovered:

        - Problems with existing system
        - Opportunity to meet new need
        - Organizational direction
        - Names of key individuals
        - Values of organization
        - Special information processing circumstances
        - Reasons for current system design
        - Rules for processing data

Analyzing Procedures and Other Documents

- Four types of useful documents
    - Written work procedures
        - Describes how a job is performed
        - Includes data and information used and created in the process of performing the job or task
    - Business form
        - Explicitly indicate data flow in or out of a system
    - Report
        - Enables the analyst to work backwards from the report to the data that generated it
    - Description of current information system

Modern Methods for Determining Requirements

- Joint Application Design (JAD)
    - Brings together key users, managers and systems analysts
    - Purpose: collect system requirements simultaneously from key people
    - Conducted off-site
- Prototyping
    - Repetitive process
    - Rudimentary version of system is built
    - Replaces or augments SDLC
    - Goal: to develop concrete specifications for ultimate system

In structured analysis there are three orthogonal views:

The functional view, made up of data flow diagrams, is the primary view of the system. It defines what is done, the flow of data between things that are done and provides the primary structure of the solution. Changes in functionality result in changes in the software structure.

The data view, made up of entity relationship diagrams, is a record of what is in the system, or what is outside the system that is being monitored. It is the static structural view.

The dynamic view, made up of state transition diagrams, defines when things happen and the conditions under which they happen

2.7 Data Flow Diagrams

System analysts use process models (i.e. data flow diagrams, DFDs) to show information flow and processing in a system. The model usually starts with a context diagram showing the system bubble surrounded by the external environment identified by external entities. Data flows bring information to and from the system process. A process can explode to a child diagram that presents its details using data stores, data flows and sub processes. The diagram leveling process allows complex systems to be easily partitioned into a stack of simple diagrams with rigorous balancing of information between levels. Information structures are defined in an associated data dictionary.

Structure Charts

Structure charts show module structure and calling relationships. In a multi-threaded system, each task (thread of execution) is represented as a structure chart. Large structure charts are leveled into a stack of connected diagrams.

State Models

State models include diagrams and tables that show the significant states in a system, events that cause transitions between states and the actions that result.

Task Diagrams

- Task diagrams show threads of execution and the real-time operating system services like queues, event flags and semaphores that connect them in a multi-tasking environment. Each task can be associated with its structure chart representation.

2.8 COMPUTER AIDED SYSTEMS

With our INCASE computer-aided software engineering tool, EDS can provide your business with the advantage of systems engineering automation, resulting in significant gains in quality and implementation timeliness. Our expertise in CASE-related systems projects, combined with our commitment to deliver legendary services to our customers, provides your business with a powerful advantage. We can help your business operate more effectively by increasing your ability to respond to market changes—your ability to benefit from new opportunities will be enhanced by greater efficiencies in your information systems. For more than 30 years, EDS has helped businesses plan, develop, integrate and manage data processing and information systems.

With EDS and INCASE, you can maximize the effectiveness of your information assets, and you can make the best use of information technology as a tool to help your business take advantage of change. ***Achieve the benefits of higher quality systems*** We measure the quality of information systems according to how well they meet the needs of your business. At EDS, our people are customer-focused, constantly striving to meet and exceed the expectations of our customers.

INCASE uses a data-centered approach to ensure that your systems are not only delivered on time but also provide you with features that conform to your business requirements. Quality information systems can be developed in the most efficient manner only with the help of systems engineering automation.

With INCASE developed systems, your company's competitiveness is enhanced in a variety of ways:

Delivered systems perform as expected, allowing your business to use them to their fullest extent from the first day of implementation.

SELF-ASSESSMENT QUESTIONS: I

ANSWER THE FOLLOWING QUESTIONS

1. Design, distribute, and analyze questionnaires to determine ____________.
2. What are the types of samples?
3. What is a data flow diagram?

UNIT QUESTIONS

1. What is a fact-finding technique?
2. Explain the structured analysis strategies.
3. Explain the data flow diagrams with examples.
4. Describe the computer-aided systems in detail.

RECOMMENDETION FOR FURTHER READINGS

1. System analysis and design, Elias M.Awad.
2. System analysis and design, Lee.

ANSWERS OF SELF-ASSESSMENT QUESTIONS

1. System requirements.
2. Types of Samples
   - Convenient
   - Random sample
   - Purposeful sample
   - Stratified sample

3.Data flow diagrams are used to show the flow of information and possessing in a system.

NOTES

UNIT III THE ANALYSIS TO DESIGN TRANSITIONS

UNIT III THE ANALYSIS TO DESIGN TRANSITIONS

3.1 Specifying Application Requirements

Transformations:

Object-oriented design methods exploit continuity while proceeding from a declarative to a procedural point of view. This process may be seen as a series of transformations from the inputs to the outputs of the design process. The most important input considerations *from* analysis are of the same general form as those described in Chapter 2 as analysis *inputs*. While the headings are the same, the details are substantially more refined, and reflect the products of OOA activities.

Functionality:

The purpose of the system, as described by (multiple) declarative models of objects, classes, relations, states, transitions, interactions, etc.

Resource:

The computational substrate on which the system will be built.

Performance:

The expected response times of the system.

Miscellaneous:

Auxiliary constraints including:

Software quality requirements concerning reliability, modularity, safety, cohesion, testability, understandability, reusability, and extensibility.

Lifecycle requirements for system evolution, demanding design allowances for reimplementation, repair, extension, and related adaptations necessary for coping with future requirements.

Compatibility requirements governing interactions with other systems, subsystems, and components (most typically non-OO ones) through constrained interfaces.

3.2 Specifying Application Requirements

We describe design activities as a series of transformations. However, we cannot prescribe fully algorithmic transformation schemes. The spectrum of transformations includes a few utterly mechanical translations, some involving guided (but not fully predetermined) series of refinements, some providing a wide range of options that must be chosen using situation-specific criteria (this is the most typical case), and some for which we can only provide general advice about how to connect initial and final conditions. The structure of these transformations relies on criteria common to the design of *any* transformational process, including:

- Group all transformations into meaningful, tractable phases with well-defined inputs, outputs, and separation of concerns.
- Obey logical dependencies. Do not schedule a step until its prerequisites are complete.
- Keep downstream options open. Avoid premature commitments to nonessential details.
- Operate on the most general representation possible for any transformation, thus minimizing redundancies resulting from similar operations on special cases.
- Output refinements and restructurings using the same representational framework as their inputs.

For example, requirements may be handed to a designer in sizable chunks, or even all at once. However, because of their intrinsic dependencies, design activities attempting to deal with these requirements must be in part sequential. It is impossible to assign resources to objects and manage their use until resource demands are at least approximately determined by establishing representational and computational properties. It is similarly impossible to address performance issues until these mappings are known. Sequentially does not, however, imply that *all* activities within one sub phase should be performed before all in the next.

While we use these criteria for guiding the overall design process, they also govern the architecture of just about any transformational system. Examples include compiler design, simulation system design, and computer vision processing system design.

3.3 Phases and Activities

The three major categories of input from analysis subdivide focal design issues in a natural way. We can expand on these groupings to better characterize them from a design perspective:

Functional design:

Definition of representational and algorithmic properties of classes obeying the declarative constraints specified within OOA models.

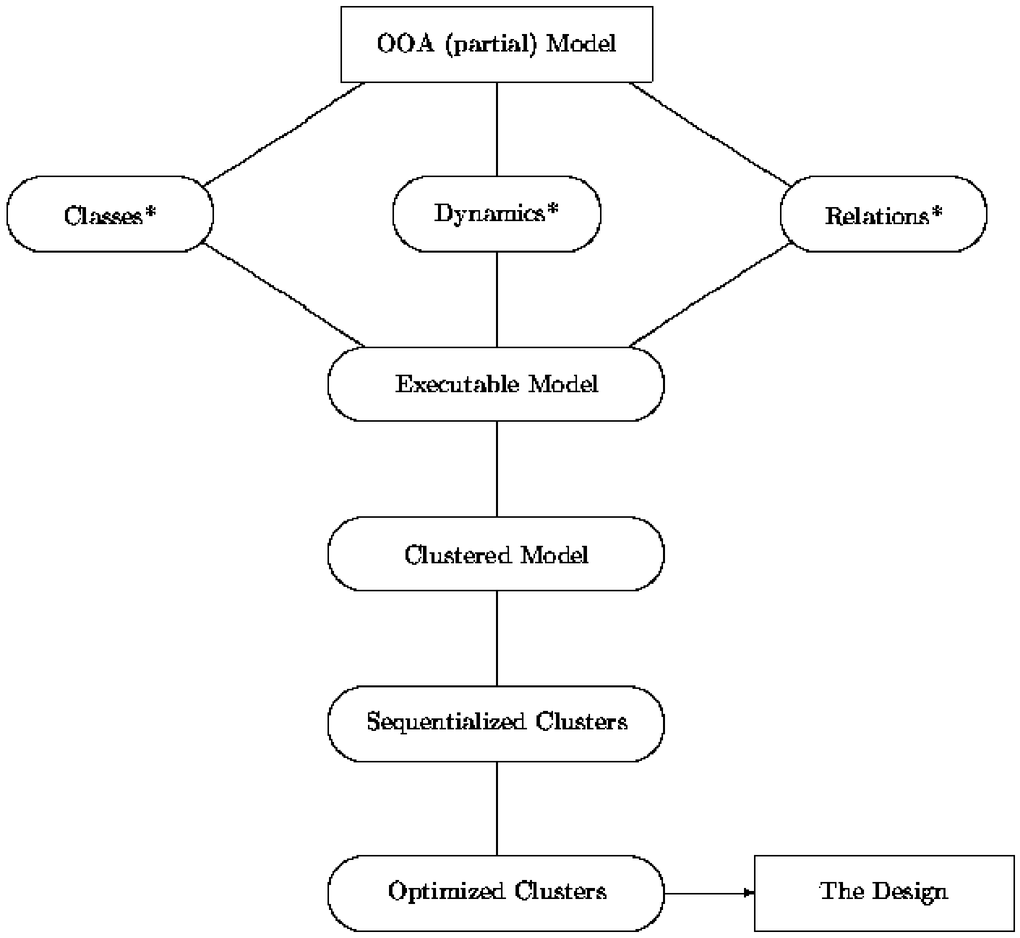
Physical design:

Mapping of objects to processors, processes, storage, and communication channels, along with design of facilities to manage these resources.

Performance design:

Reconciliation of functionality and resource mappings to meet performance requirements when expressed using the target implementation languages, tools, configurations, etc.

The distinctions between functional, physical, and performance phases help capture different concerns and activities within design. While this is a reasonable way of classifying well-accepted *object-oriented* design practices, it differs from traditional approaches in a number of respects. For example, there is no separation between ``coarse" and ``detailed" design. Our presentation actually begins by discussing some small granularity issues, and our performance design methods consider both in-the-small and in-the-large performance factors. Thus, these categories focus more on the goals of design than their granularities. Similarly, we do not focus on (logical) ``architectural design" *per se*. OOA models already specify much of the logical structure of systems. While many aspects of this structure may be modified during design, their general forms are established through declarative modeling.



The basic framework admits several variations. For example, functional design may be interleaved with parts of OOA by dealing with subsystems or other coarse-grained components as they become available. Design activities may be subdivided among designers working concurrently and semi-independently. Performance design and implementation activities may be similarly distributed and pipelined. Also, design of the general form, policies, and infrastructure of many physical system matters may proceed at nearly any point.

3.4 Objectives in Designing Information Systems

Identify agents and their operations

• Types of people

• Types of operations

Identify agent/operation relationships

• Who does what with whom/what?

Identify data requirements

• What information is involved?

• What fields does the database require?

• what types are the fields?

• What formats should be used?

• General formats of database files

Different design cycles

Dataflow diagrams

Structure charts

3.5 Design of Computer Output:

- The question of output doesn't just apply to screen and paper. You should consider it for all output, down to the humblest bleeps and whistles. Remember, the poor bloody user will be faced with your output day in, day out. You may have tested the program twenty times, but he/she will have to run it thousands of times - and a misplaced "ding", or an unnecessary "Are you sure?" box will cause massive irritation.
- The more unusual sides of output include one-off things like presentations, for which output mechanisms like Overhead Projections (OHP) and PowerPoint are appropriate. They may be one-offs, but they are usually important. In a presentation you are invariable selling something (yourself, usually), so make a good impression.

- And let's not forget the Internet, specifically the World Wide Web. (You don't know the difference? The WWW is the graphical all-singing-all-dancing bit which most people think is all there is to the Internet. The rest of it is boring text next works, file transfer protocols and bulletin boards - yawn!) Let's face it, the first most people will see of your company is its web page, and you know what they say about first impressions!
- To a lesser extent, photography and video. I have seen some cringe-making publicity films in my time!

Six criteria to be considered when designing output

- What’s the purpose of the output? Don’t produce the output simply because you can! Every piece of output should be justified - don't write some report on it, just think to yourself "Why does the user need to see this?"
- For whom is the output intended? Consider different forms of output for different users. Some people like paper, others prefer seeing it on a screen.
- To whom is the output to be distributed? (Another side of point 2 above?) This really influences how many copies of the output are created. Of course, with electronic output you can make as many copies as you like with no extra effort.
- How much output is needed? Don’t produce masses - it will be ignored!
- When is the output required? If it arrives too late, it is useless. If it arrives too early, it will be ignored (and forgotten about, probably until it is too late. You just can't win!)
- What is the most appropriate output method? Normally this means paper vs. screen (possibly with printout facilities).

3.6 Designing Printed Output

- Ink jet printers are cheap and cheerful. They typically cost about £300 and can cope with either A3 or A4 paper. They run on ink cartridges, with an expected lifetime of over 1000 copies (depending on how much printing there is on average on each page). Color ink jets are commonplace, and they are fitted with black-ink cartridges and triple-color cartridges - the colors being yellow, cyan (similar to turquoise) and magenta (a rather apoplectic pink color), which can be combined with black ink in various combinations to form any color except white (but since the paper is white, that's not a problem!)

- Laser jets are expensive but high quality, and often double up as photocopiers, scanners and fax machines. They run on toner cartridges, toner being a black powder similar to soot. The laser doesn't actually burn the image into the paper - it merely charges it with static electricity at all the points on the paper where a black mark is to appear. The static electricity makes the toner stick to the paper at that point (like rubbing a balloon against your jumper). The paper is then heated to melt the toner onto the paper - which is why paper coming out of a laser printer is always warm.
- Very old dot matrix printers. Poor quality, noisy, but dirt cheap. They use ribbons and work similarly to typewriters, with little prongs that hit the ribbon and transfer the ink to the paper. These are the oldest form of printing (short of getting monks to write the documents out by hand) and probably the cheapest. Although dot-matrix output is the worst quality, it is often good enough for many purposes (memos, archive storage etc.)

3.7 Designing Visual Display Output

- This doesn’t necessarily mean a fully interactive computer program. It could be a simple dedicated system. For example, the London Underground has a series of dedicated terminals with touch-sensitive screens for passengers to find out information. There is no keyboard, and the users can only select options from menus (by pressing on the screen).
- Different degrees of interactivity. This ranges from no interactivity at all (for example, a rolling display in a shop window) to a fully fledged computer program running on an ordinary computer (where the user can do anything, including stopping the program and starting another one). As more interactivity is introduced, the problem gradually turns from one of output to one of input and output.
- Takes advantages of all the facilities of computer processing e.g. databases. A computer program can process raw data and turn it into proper information. Computer screens can also decide which information is relevant at any one time. A good example is series of screens in the Space Shuttle cockpit. The flight computer (well, there are five of them, actually) decides what information is needed at any time, and just shows that on the screen. The pilot is given the choice of selecting any other information at any time, of course, but this way there is no chance of missing something important.

- Unless a printout facility is provided, there is no permanent record. This often isn't a problem. You probably don't need a printed copy of every piece of data. On the other hand, if a print facility *is* included, there is the risk that users will simply print everything and just rely on the printouts. By refusing to include a printout facility, the programmers are forcing users to update their attitudes towards data - or simply encouraging them to ignore the program in favors of another.
- You have a choice: Recorded speech vs. Synthesized speech. Recorded speech sounds natural, but you are limited as regards the number of words and phrases you can include. This is fine in situations like announcements at railway stations, where the structure of the announcements is severely restricted. The alternative, synthesized speech, involves the computer producing words from individual sounds (called ***phonemes***). These phonemes are themselves produced by rules concerning frequencies present in different sounds. Unfortunately, as anyone what has listened to Stephan Hawking will tell you, the results sound like a bronchial dalek on a bad day. However, you can, in principle produce any words, even new words.

  Speech synthesizers contain rules which tell it how to turn letters of the alphabet into strings of phonemes ('cat' into phonemes /k//a//t/), but English is notoriously unreliable about spelling rules. The results of asking a speech synthesizer to pronounce "Power mowers" or "We thoroughly plough through the dough" are quite amusing. Recently, work has been done by Terry Sejnowski in using a neural network to pronounce English text, and it can learn the irregular rules of English letter pronunciation quite easily.

  Synthesized speech can be altered for different purposes. For instance, decreasing the general frequencies of phonemes lowers a voice, and makes it more "masculine". A compromise between synthesized speech and recorded speech is to record a series of phonemes spoken by a human (they are "extracted" from entire words in the recording) of varying intonations (rising, falling tone etc.) and to string them together to form words. The results tend to be better that pure synthesized speech, but still sound rather drunken.
- Speech is immediate and intuitive. It requires no training - anyone who isn't deaf can understand it. Speech can be augmented with sounds, music etc.
- Good for individuals. Using headphones, each individual can have his/her own soundtrack independent of the others. This is the approach

used in stately buildings where tourists are given an audio guide and headphones, and each can move at his/her own pace. An audio tone is the signal to the user to move to the next point on the tour.

- Speech is good for hands-free environments, or when the users have to turn their heads (e.g. stacking items in a factory). Unlike using a computer screen, the user doesn't always have to look the same direction. The ability to receive computer output while moving around or turning reduces repetitive strain injury and eye strain.
- Audio output is impermanent. Like display screens, once the computer moves on to something else, there is no record of the output. This doesn't invalidate it as a communication mechanism, of course - spoken words are equally impermanent, but I doubt that anyone would think that makes talking a worthless activity.

3.8 Capturing data for input

- CD-ROM stands for Compact Disc, Read Only Memory, indicating that they are recorded by the programmer and then can only be read by the user. The user can't record on or change the contents of the CD-ROM in any way. They are identical to audio CDs except that they hold digital computer data - typically 640 Megabytes, which is about 450 times the amount of data held by a floppy disc. In recent years special CD-ROMs, called CD-RWs (CD Read Write), have started to appear, which can be rerecorded over, like floppy discs. They require a special CD-Writer drive to do this - it can't be done using a normal CD-ROM drive on a computer.
- The replacement to CD-ROMs (and CD-RW, I dare say) is DVD - Digital Versatile Disc. These used to be called Digital Video Discs, as they first started to appear as ways of storing vide images. They can store many times more than a CD-ROM, typically 8 GB of data. Up to now, they have generally been used for holding feature films. As the years progress, the technology will become more flexible, and, no doubt, read-write DVDs will appear.
- Both CD-ROM and DVD have massive capacities when compared with floppy discs. This is ideal for such data-hungry software as multimedia applications, graphics, sound etc. The use of CD-ROMs has allowed software to become a lot more picture-based, with thousands of still images and snippets of video footage.
- Although initially expensive, they are now relatively cheap to produce (the typical cost is 20p when produced on a large scale). There is an initial cost for the CD-ROM writer, but these have dropped in price

from several thousand pounds a couple of years ago to a couple of hundred pounds or so.

- Typical uses for CD-ROMs include advertising, visual games, archive data storage of massive amounts of data etc.

E-mail, Web pages, (Fax)

- These are global - they can be viewed anywhere in the world at (close to) zero cost, providing you can lay your hands on a computer with an Internet connection.
- They can be updated very easily by the person who created them and posted them on the Internet. The updated files are instantly available.
- By the same token, it is (supposed to be) impossible for any unauthorized person to alter the web page. However, hackers have been known to alter web pages and change hyperlinks.
- Pages on the World Wide Web (the WWW - the graphical part of the Internet that you are looking at now) are written in a language called Hypertext Markup Language (HTML). You can see a sample of HTML by selecting the View Source option from the View menu of your browser. However, it is not necessary to understand raw HTML to create a web page. There are many programs which provide a user-friendly front end to the web page design process, allowing you drag and drop items on a screen. The program then turns your page into HTML. Such programs include Dream Weaver (the industry standard at the moment), AOLPress (which can be downloaded free at the moment from the AOL website) and Microsoft's Front Page Editor.
- Web pages can be made interactive, using ***forms***. These are boxes that allow viewers (it seems inappropriate to call them "users") to enter data and send it back to the company or person that created the web page. They can also be linked to an E-mail facility, so that the viewer of a web page can click on an icon and instantly send an E-mail to the web page creator.

Moving up the scale, the computer language Java was designed to run programs on the Internet. These programs are called ***applets*** and they are embedded in web pages. However, an Internet browser has to be "Java enabled" for it to run applets. Java has a younger brother, called JavaScript, which is easier to embed in a web page. While not as powerful as Java, it does give a degree of interactivity.

The latest addition to Internet programming languages is Shockwave, which allows multimedia applications to be transferred to the Internet. Many programs such as Microsoft Word and Powerpoint have a facility so that documents and presentations can be turned into HTML code ready to be uploaded on to the World Wide Web.

- There is still a great deal of resistance to "new-fangled" technology such as web pages. Only a small fraction of people is connected to the Internet (or have access to it at places such as Internet cafés), although this proportion is increasing all the time.

The intended user influences the choice of output

- Who will see the output? I'm sure you would design output very differently if you knew that the boss was going to see it than the way you would design it if you knew only the secretaries would see it.
- Different users will have different needs and different equipment (e.g. more/less powerful computers, different software). Software that produces output may require high speed processors, for instances graphs of stock market data that are updated and displayed constantly. A slow computer would lead to the graphs only being updated, say, once a minute, which would lead to a disadvantage for that particular stock trader.
- Different standards can be applied for different users e.g. a report to the board should be high standard, whereas an internal memo can be draft quality. This also applies to the standard of English used, the necessity for spell checking etc.

For how long is the output to be stored?

- Output can be archived, or for immediate use only. It may have a "sell-by" date (e.g. valid for a month, then useless rubbish after that).
- If output is to be archived, consider printout, magnetic tape, or (better still) CD-ROM/DVD.
- If output is to used and discarded almost immediately, or if it can be recreated very easily, consider computer screen/web site.

Long-term output

- Printouts are easy to use (as long as you can read!) but take up a great amount of space and can be difficult to manage ("drowning in paper"). A good example of this is the Yorkshire Ripper enquiry in the 1970s and 1980s held by Yorkshire and Humberside police. The serial killer's identity was contained inside the paper-based system for many years before he was caught. There was no central computer system that could collect and correlate all the data, with the result that his file was often overlooked (e.g. out on someone's desk when someone else needed). It was partly as a result of this case that the central Police National Computer system was set up.

- Magnetic tape ("Mag. Tape") has a very large storage capacity, but it can sometimes be corrupted. It is difficult to load into the mag. tape drive reader, and is a **serial device**, meaning that the tape has to be read through inch-by-inch before the reader reaches the right place. It is not suitable for data that requires a lot a dodging around.

  Magnetic tapes are usually used for **batch processing**. This involves keeping account of all the records that require updating, sorting them into order, and then running through the magnetic tape from one end to other updating the records as it comes across them. This is suitable for tasks such as calculating payrolls, where thousands of records may need updating. Often computer operators would fit the magnetic tapes to the drives on Friday, leave the computer running all weekend, and return on Monday morning to find the job finished.
- CD-ROM/DVD is the most efficient way of storing massive amounts of data in a tiny space. It has been estimated that a CD-ROM can hold the entire text of the complete works of Shakespeare 450 times over, although data such as pictures, sound and video clips require much storage. That is why a 640 Megabyte CD-ROM can only hold 74 minutes of CD quality sound. 640 Megabytes translates roughly to 640 million characters (letters of the alphabet or spaces), which compares to about 1.5 million characters for a floppy disc, and 2,000 characters for a piece of paper. It is also the reason that video clips from a CD-ROM tend to appear in a very small window on the screen - that takes up less space on the CD-ROM.

Back-up copies

- For electronic media, ensure that up-to-date back-up copies are also present. It is an irksome task that most computer operators hate doing. If they fail to make a back up, they may live to regret it, when some twerp spills coffee over the magnetic tape and ruins it. I have been in that position myself - I lost nine months irreplaceable work!

  Special back-up drives exist with large capacities. They can be set up to make the processing of back-ups as easy and unintrusive as possible. They can be set up to back the data up automatically once a week (or at some similar interval), just back up the newest data files (created or edited after a certain time or date), and to maintain multiple back-up copies (the so-called "grandfather-father-son" method where the last three back-ups are kept and the one before that is recycled to form the next back-up).

- Although CD-ROM/DVD is permanent (non magnetic), they are easy to break! Being non-magnetic, they are immune to magnetic fields (such as the ones at airport security check-ins). However, it is recommended that you don't press too hard on a CD-ROM, scratch it or write on it using anything other than a soft-tipped pen. For further safety, keep the CD-ROM in its plastic case when it is not being used.
- In the future, will the technology be available to read these electronic media? This isn't really a problem with CD-ROM/DVD, as any new technology that comes to replace it will have an overlap period. During that period, the technology will be available to transfer the data from CD-ROM to the new storage medium.

Where is the O/P needed?

- Does the output need to be ***portable***? If so, consider using media such as floppy discs. They are small, dirt cheap (about 10p per disc), and more widely usable than CD-ROM. All computers are fitted with a floppy disc drive, as compared to only most computers having a CD-ROM drive. They also have fairly large capacity compared to paper.
- The Internet provides global availability provided one could get to a computer with an Internet connection!
- If the output does not need to be portable, consider paper printouts or magnetic tape. They can be stored in a "library" and accessed when necessary. Libraries of CD-ROMs can be accessed remotely on a series of terminals via a network, and server software allows several people to access the same CD-ROM simultaneously.
- Reference journals, microfiche etc. are found in universities. Microfiches were a pre-CD-ROM attempt to miniaturize data to reduce storage space. They usually contain data such as newspaper articles, abstracts for academic papers etc. shrunk down to miniscule size (similar to the sort of microdots that spies use, only not shrunk down so far). Microfiches require a special reader to magnify the images. Of course, CD-ROMs are gradually making such devices redundant, although it will be many years before the last ones disappear.

  Online libraries are starting to appear on the Internet. An example is the A.C.M. digital library, which contains articles from recent issues of various academic journals. These libraries are generally accessed on a pay-per-view basis, i.e. you have to give your credit card details and you are charges so much per minute.

How frequently will the O/P be needed?

- Paper copies eventually wear out/fade. This is especially true if they are kept exposed to sunlight. They are like very slow vampires! Of course, this isn't a problem if they can be regenerated from computer software, but if they can, why keep the printouts in the first place? If they can't be regenerated, it can be hard, expensive or time consuming to make copies, by photocopying them.
- Electronic media are potentially "immortal". They never fade or wear out, and an unlimited number of copies can be made. This raises its own set of problems - How do you keep track of copies? How do you ensure that they are up-to-date? How do you ensure the copies are authorized? (for example "Data piracy" or "Software piracy"). They take up less space than paper copies and use less resources. Specifically, they use less paper! I'm all in favour of saving paper. After all, it doesn't grow on trees, you know!

SELF-ASSESSMENT QUESTIONS: I

ANSWER THE FOLLOWING QUESTIONS

1. What is functionality?
2. What are various categories of design?
3. Design activities may be subdivided among ________ working concurrently and semi-independently.

UNIT QUESTIONS

1.What are the various objectives in designing information systems?

2.Explain the design of computer output.

3.What are the special features of visual data output?

4.Write notes on input validation.

RECOMMENDETION FOR FURTHER READINGS

1. System analysis and design, Elias M.Awad.
2. System analysis and design, Lee.

ANSWERS OF SELF-ASSESSMENT QUESTIONS

1. The purpose of the system, as described by (multiple) declarative models of objects, classes, relations, states, transitions, interactions, etc.
2. * Functional
   * Physical
   * Performance

3. Designers.

NOTES

UNIT IV: DESIGN OF ONLINE DIALOGUE

UNIT STRUCTURE

UNIT IV: DESIGN OF ONLINE DIALOGUE

4.1 Interface Definition

An interface is a device or a system that unrelated entities use to interact. According to this definition, a remote control is an interface between you and a television set, the English language is an interface between two people, and the protocol of behavior enforced in the military is the interface between people of different ranks.

An interface is the boundary between two non-miscible materials. There are interfaces between liquid-solid, liquid-liquid, liquid-gas, solid-solid, and solid-gas – but not gas-gas. An interface between liquid-gas or solid-gas is called a surface.

Importance

We are surrounded by interfaces and interfacial phenomena in our everyday lives – such as using shampoo or deodorant, preparing meals, washing our clothes and dishes, and wearing contact lenses. Forming a solid-solid interface by joining two materials together with adhesive or glue is something we take for granted – from postage stamps to airplanes. All of these familiar objects have numerous interfaces that are designed and controlled through interfacial science and technology.

4.2 Design of Files and Auxiliary Storage devices

To understand the problem with the first file system, we have to define some terms:

- **I-Node:** An inode is quite simply a data structure that stores a link to the file data, the file name and metadata (such as file size and creation date). These inodes are what define the structure of the drive. Have you ever used a card catalog in a library? This is pretty much the same idea. Actually, an inode is almost exactly like a card in the catalog.
- **Block:** A block is the software-level abstraction of a sector. Basically, if a disc has a sector size of 512 bytes and yet the OS wants a 1024 sector size, they just call it a block and not a sector. The words block and sector are used interchangeably in a lot of documentation and can kind of be interchangeable in your mind as well.
- **Superblock:** The superblock is usually located at the front of the file system and defines it. This block contains various statistics about the file system, such as size, a magic number that supposedly uniquely identifies the file system type and, in the original Berkeley file system, a list of the partition?s inodes.

- **Data Blocks:** This is the actual data, strangely unimportant in file system design. (and this article)

So, what was the problem with the Berkeley file system? Simple. All the inode information was stored in the superblock. This works fine for small drives, but picture in your mind a large drive. This would mean that the read head would have to go to the beginning of the drive to read an inode, then to the data blocks to get the file data, then back to the inode section, then back to the data and so on. This is called thrashing and is about as inefficient as file systems get. The good people at Berkeley set out to solve this problem in creating their new faster file system. They were exceptionally creative about the name and ended up calling it the Fast File System.

4.3 Auxiliary Data Storage

Most digital computers store data not only in their RAM memory but also on auxiliary storage units. Here data and programs can be stored much like a file cabinet, not only for easy retrieval, but also to store data and programs that are too large to fit into the random-access memory at one time. These storage devices also offer a more permanent and secure method for storing programs and data compared to RAM memory, but much like RAM, offer a direct access to the data.

Floppy disks, hard disks, magnetic tape, and optical disks are examples of auxiliary storage devices. Floppy-disk drives (removable magnetic disks) can store a few million bytes of data on one disk and are used primarily in laptops and PCs. Hard disk drives are non removable magnetic media and are used with all types of computers. Compared to floppy disks, they access the data very quickly and can store significantly more information: from millions (megabytes) to billions (gigabytes) of bytes of data.

Magnetic-tape storage devices are much like audiocassette tapes. These devices are usually used on main-frame computer systems to handle high volumes of data. Compared to disks, tape drives access data very slowly and only sequentially. Nowadays, the role of the tape drive is limited to backing up or duplicating the data in the hard disk drive to protect the system against loss of data during power failures or computer malfunctions. The capacity of these devices ranges from few megabytes to gigabytes.

Optical disks are non-magnetic auxiliary storage devices that resemble audio compact disks. Unlike magnetic media, the data are encoded on a disk as a series of pits and lands that can be read by a laser. Current technology allows about 600 megabytes of information to be stored, including reference text, computer programs, pictures, sound, and simple motion pictures or animation. Most common are the read-only CD-ROM (or compact disk, read-only memory) disks on which manufacturers have pre-recorded the information onto a CD-ROM. Recordable CD-ROM disks, called WORM (write-once/read-many) are becoming an affordable alternative to tapes and hard disk, primarily for archival storage purposes. A single, small CD-ROM disk can hold more information than 1,000 floppy disks.

4.4 Data Structure Diagrams

A data structure diagram (DSD) is the result of a process of hierarchical decomposition of a complex data area, which is subdivided as far as possible (and reasonable). DSDs are hierarchical tree diagrams depicting "may consist of" relationships between data items if read from top to bottom and following the connecting lines (Diagram 1). The boxes in the diagram may represent

- Intermediate complex data items which are further subdivided in the diagram
- Fully atomized data items, represented by an attribute, which can be directly used as a field definition in a database table.
- References to complex data items detailed elsewhere: Instead of listing attributes, a reference to an entire entity may be made (entities may be depicted by a data structure diagram with only one row of data items - attributes - below the header item). Another possibility to make a reference to several data items is to include a part, i.e. a reference to a separate data structure diagram.
- Conditions: The "consists of" relationship may be modified by conditions written in boxes which are marked by an abbreviation on the right hand top of the box defining one of the following conditions:
  "IF": Data items below this box are to be read only if the condition is satisfied
  "EXCL" (= exclusive alternative): Same as "IF", but several such conditions exist which are mutually exclusive
  "LOOP": Data item is repeated as many times as indicated in the condition.

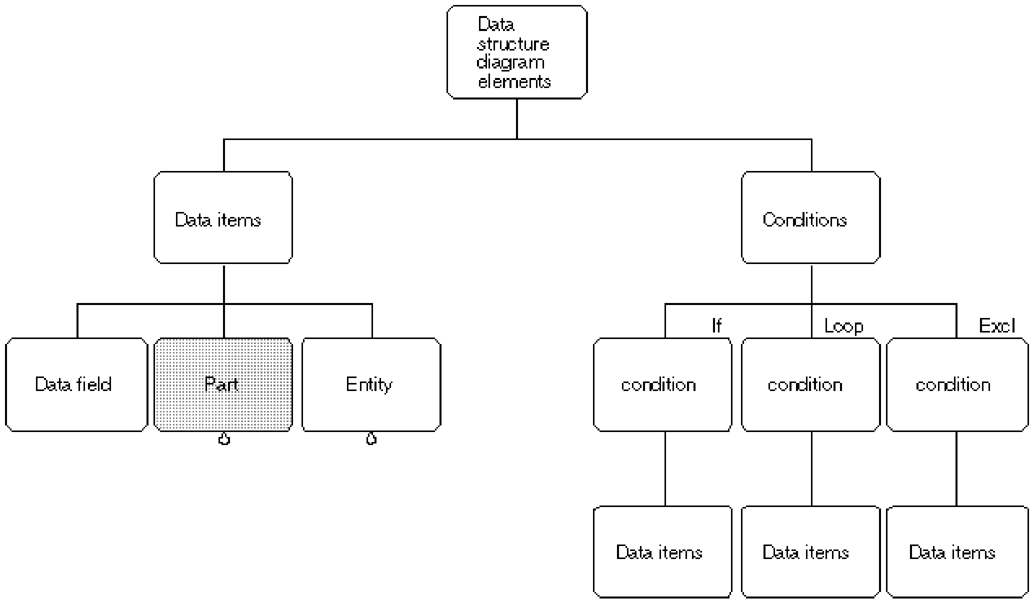


Diagram 1: Data structure diagrams

Attributes are marked by the abbreviation "Attr." on the right hand top of the box. A little arrow pointing to the bottom of a shaded data box indicates that it represents a part, a little arrow pointing to the bottom of an unshaded data box indicates entity. The name of the entity is given in parenthesis in the box.

DSDs may span several entities. Fully resolved DSDs (without references to complex data items) are related to "views" in a relational database system. Loops in the DSD represent a one-to-many relationship if read from the top.

4.5 Methods of File Organization Introduction

File organization is the methodology, which is applied to structured computer files. Files contain computer records, which can be documents, or information, which is stored in a certain way for later retrieval. File organization refers primarily to the logical arrangement of data (which can itself be organized in a system of records with correlation between the fields/columns) in a file system. It should not be confused with the physical storage of the file in some types of storage media. There are certain basic types of computer file, which can include files stored as blocks of data and streams of data, where the information streams out of the file while it is being read until the end of the file is encountered.

We will look at two components of file organization here:

1. The way the internal file structure is arranged and
2. The external file as it is presented to the O/S or program that calls it. Here we will also examine the concept of file extensions.

We will examine various ways that files can be stored and organized. Files are presented to the application as a stream of bytes and then an EOF (end of file) condition.

A program that uses a file needs to know the structure of the file and needs to interpret its contents.

Internal File Structure

Methods and Design Paradigm

It is a high-level design decision to specify a system of file organization for a computer software program or a computer system designed for a particular purpose. Performance is high on the list of priorities for this design process, depending on how the file is being used. The design of the file organization usually depends mainly on the system environment. For instance, factors such as whether the file is going to be used for transaction-oriented processes like OLTP or Data Warehousing, or whether the file is shared among various processes like those found in a typical distributed system or standalone. It must also be asked whether the file is on a network and used by a number of users and whether it may be accessed internally or remotely and how often it is accessed.

However, all things considered the most important considerations might be:

1. Rapid access to a record or a number of records, which are related to each other.
2. The Adding, modification, or deletion of records.
3. Efficiency of storage and retrieval of records.
4. Redundancy, being the method of ensuring data integrity.

A file should be organized in such a way that the records are always available for processing with no delay. This should be done in line with the activity and volatility of the information.

4.6 Types of File Organization

Organizing a file depends on what kind of file it happens to be: a file in the simplest form can be a text file, (in other words a file which is composed of ASCII (American Standard Code for Information Interchange) text.) Files can also be created as binary or executable types (containing elements other than plain text.) Also, files are keyed with attributes, which help determine their use, by the host operating system.

Techniques of File Organization

The three techniques of file organization are:

1. Heap (unordered)

2. Sorted
    1. Sequential (SAM)
    2. Line Sequential (LSAM)
    3. Indexed Sequential (ISAM)
3. Hashed or Direct

In addition to the three techniques, there are four methods of organizing files. They are **sequential, line-sequential, indexed-sequential, inverted list** and **direct or hashed access** organization.

Sequential Organization

A sequential file contains records organized in the order they were entered. The order of the records is fixed. The records are stored and sorted in physical, contiguous blocks within each block the records are in sequence. Records in these files can only be read or written sequentially. Once stored in the file, the record cannot be made shorter, or longer, or deleted. However, the record can be *updated* if the length does not change. (This is done by replacing the records by creating a new file.) New records will always appear at the end of the file. If the **order of the records** in a file is not important, **sequential organization** will suffice, no matter how many records you may have. Sequential output is also useful for report printing or *sequential reads* which some programs prefer to do.

Line-Sequential Organization

Line-sequential files are like sequential files, except that the records can contain only characters as data. Line-sequential files are maintained by the native byte stream files of the operating system.

In the COBOL environment, line-sequential files that are created with WRITE statements with the ADVANCING phrase can be directed to a printer as well as to a disk.

SELF-ASSESSMENT QUESTIONS: I

ANSWER THE FOLLOWING QUESTIONS

1. What is an interface?
2. What are various types of file organization?
3. In the COBOL environment, line-sequential files that are created with WRITE statements with the ________ phrase can be directed to a printer as well as to a disk.

UNIT QUESTIONS

1. Define an interface.
2. Explain the concepts of file organization.
3. Describe the internal file structure in detail.
4. Explain elaborately the data structure diagrams.

RECOMMENDETION FOR FURTHER READINGS

1. System analysis and design, Elias M.Awad.
2. System analysis and design, Lee.

ANSWERS OF SELF-ASSESSMENT QUESTIONS

1.An interface is a device or a system that unrelated Entities use to interact.

2. * Heap
   * Sorted
   * Hash

3. Advancing.

NOTES

UNIT V SYSTEN ENGINEERING AND QUALITY ASSURANCS

5.1 Design Objectives

Background History

Recent developments in communications have led to the design and deployment of increasingly complex digital systems, which are replacing analog systems. Digital systems provide an improved image and sound quality compared to analog systems. The rapid degradation of signal quality at the edge of the coverage associated with most digital systems requires more precise coverage predictions. Finally, the advent of digital techniques has given way to new, often more complex, coverage scenarios where self-interference often becomes a major issue in system planning.

The need for advanced coverage simulation software is clearly in evidence due to the limitations in current simulation software tools for research and system design. Since our first software demonstrations in 1991, CRC has seized the opportunities to expand its experience and develop new concepts in the areas of coverage prediction and software engineering.

Design Objectives

In designing CRC-COVLAB, the requirement for a high level of code re-usability and modularity is achieved by using true object-oriented design. A high level of independence among objects is achieved by completely encapsulating each object and by using polymorphism. Any object in CRC-COVLAB is independent in a way that in relation to the other objects, it can be replaced by an object that has the same interface without modifications to surrounding elements.

Another important objective in designing this software was to meet and to exceed expectations for today and tomorrow's needs. It was achieved by deploying seamless integration with popular generic applications, such as Microsoft Office; as well as more technical applications, such as MapInfo and Visual Basic. This is performed through the use of various solutions, ranging from clipboard copy of the graphic output to direct connectivity using OLE/COM/ActiveX/DCOM support.

The features of Design Objectives

1. To know about the design loop / process.
2. To be able to individually identify a problem situation.
3. To be able to research an area then sifts through all available information from a variety of sources and applies relevant data to the problem.

4. Having developed an understanding of a problem a pupil should be able to draw up a brief or specification based upon all the information available and taking into account any constraints.
5. To be able to produce and record suggestions that may lead to a solution.
6. To be able to plan out in advance how the problem will be tackled and manage time to suit.
7. To be able to design realistic and relevant solutions.
8. To be able to relate all scientific and technological information to the design process.

5.2 Program structure charts

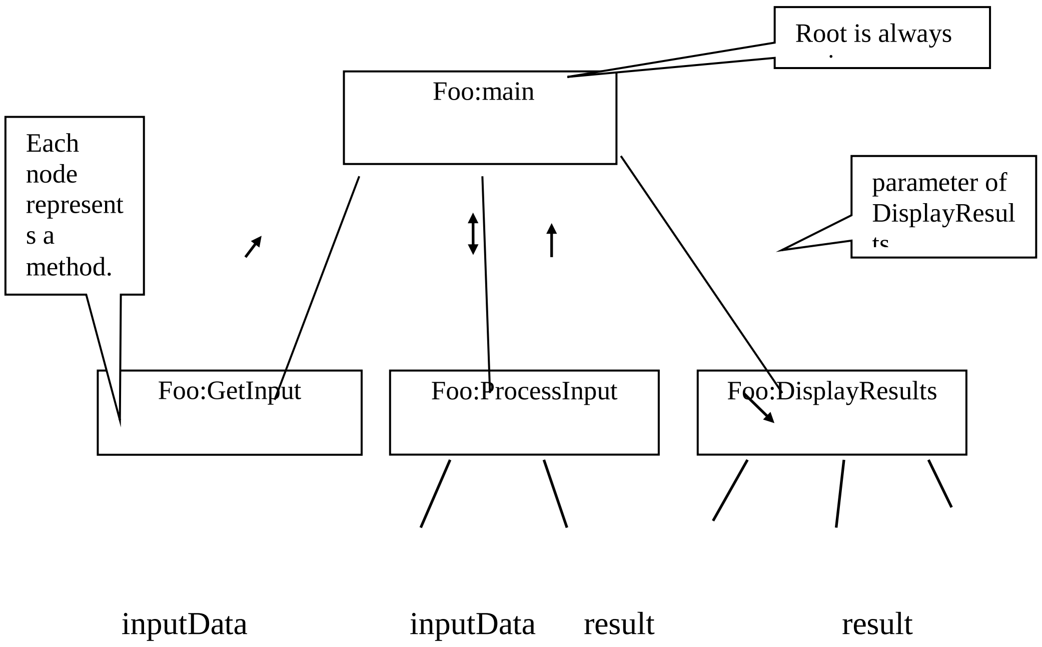
Structure charts are used to specify the high-level design, or architecture, of a computer program. As a design tool, they aid the programmer in *dividing and conquering* a large software problem, that is, recursively breaking a problem down into parts that are small enough to be understood by a human brain. The process is called *top-down design*, or *functional decomposition*.

Programmers use a structure chart to build a program in a manner similar to how an architect uses a blueprint to build a house. In the design stage, the chart is drawn and used as a way for the client and the various software designers to communicate. During the actual building of the program (implementation), the chart is continually referred to as the master plan. Often, it is modified as programmers learn new details about the program. After a program is completed, the structure chart is used to fix bugs and make changes (maintenance). It is especially important in this stage, because often the original “architect” is long gone, and a programmer new to the project uses the chart as a navigational tool into the often huge set of source code files. If you've ever experienced the joy of modifying someone else's program, you know how important this blue print can be.

The first step in creating a structure is to place *Class::main* in the root of an upside-down tree which forms the structure chart. The next step is to conceptualize the main sub-tasks that must be performed by the program to solve the problem. These sub-tasks are placed in nodes below the root, and connecting lines are drawn from the root to each sub-task. Next, the programmer focuses on each sub-task individually, and conceptualizes how each can be broken down into even smaller tasks. Eventually, the program is broken down to a point where the leaves of the tree represent simple methods that can be coded with just a few program statements.

Once the structure chart has been designed, it is used to drive the software implementation. Branches of the tree are assigned as *modules* to be programmed by various members of the development team. Often, a process called *bottom-up implementation and testing* is used to implement each module. This process will be discussed in detail later in this document.

Consider the following sample structure chart. Many information-processing applications can be modeled with a chart similar to this sample:



A Sample Structure Chart

5.3 THE SOFTWARE DESIGN

Software has become an industry in itself. Before we discuss it further, we define the type and characteristics of software for system development.

TYPES OF SOFTWARE:

Software is classified according to whether it performs internal computer functions or allows use of the computer for problem solving. The former is called system software-programs designed to control system operations (e.g., operating system and data base management systems) and system implementation (e.g., assemblers and compilers). The latter classification is called application programs, which perform user-oriented functions. Within this classification we have two groups:

1. Cross-industry application software, such as accounts receivable and payroll calculations.

2. Industry-specific software, such as a safe deposit tracking system, hospital-billing system, or airline reservation system.

In general, software can have one or more of the following attributes:

1. CONCURRENCE OF OPERATION:

Software allows simultaneous activities to take place in a computer run. For example, keying in data on the terminal takes place while the system is reading in data from disk and the central processor is operating on a separate activity.

2. RESOURCES AND INFORMATION SHARING:

Different programs share the same hardware resources. Different users may use various programs, or different programs use a centralized database. This implies multiple interfaces of a system with the outside world.

3. MODULARITY:

Software consists of segments connected in various ways, partly due to the separate functions they perform.

4. MULTIPLEXED OPERATIONS:

Some software system rotates the use of resource, such as an Input/Output device, among different users, while each user has the impression of being the sole user of the system.

The software industry, represented by time-sharing facilities management, turnkey systems, packaged software products, and cotract programming, is in the midst of an incredible boom. IBM's "unbundling" of its software from hardware in January 1970 sparked an exponential growth in the software industry. The astounding growth of the software industry is out pacing virtually all other areas. The stock prices software companies continue to rise, and new software products find immediate success. In the microcomputer environment, William Gates' Microsoft Corporation started with 2 employees in 1981. Since it developed the disk operating system (DOS) for the IBM PC late that year, it has grown to a multi million-dollar corporation serving the software needs of other computer makers as well.

Industry wide 1984 revenues were \$8.6 billion and are estimated to grow by at least 20 percent through the 1980$_s$, especially in the software products area. There are three reasons for the growth:

1. Programmer shortage. The demand/supply factor for programming is such that computer service firms are filling the gap with "canned" solutions ranging from utilities and programming to database management system and applications.

2. Hardware / software cost reversal. Although the price of software the relative to hardware is rising, the actual price is decreasing, considering the power and performance of good software. With software geared for the mass

market, application developers have scale down hardware requirements more toward the personal computer, while providing equal capabilities and lower prices than the software a decade ago.

3. Economies of scale. Assembling and distributing specialized computer solutions for a vast customer base and potentially cheaper than users acquiring their own skills and facilities to do the same. Knowledge of the computer and off software is helpful as a basis for selecting hardware / software. Today's maturing market means a wide choice for the user. Searching for the best product requires specialized knowledge and a serious approach. This is when an experienced analyst or outside consultant to a successful installation.

5.4 Training

Organizations worldwide are discovering the many advantages of Web-based training over traditional training methods, obtaining significantly improved training results at substantially reduced training costs. Now, read about these cases and find out how you can too in the Web-Based Training Cookbook.

Multimedia training expert Brandon Hall schools you in Web-based training principles, introduces you to the three main types of courses, and arms you with the know-how and tools you need to create the type that's best for your organization. He covers all the bases for managers, trainers, and Webmasters, including how to:

- Estimate costs, conduct ROI studies, write proposals, and build a business case for Web-based training
- Develop a technology plan and create a project team of instructional designers, content experts, and Webmasters
- Convert existing curricula and design new courses specifically for the Web and evaluate existing Web-based training courses
- Build both static and interactive multimedia testing and training sites.

The CD-ROM provides demos of an online testing program, an HTML training site, and a sample multimedia program from a custom developer. Also provided is source code from several Web programs and links to the sites discussed in the book.

5.5 Conversion

In computer science and information theory, **Huffman coding** is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix-free code (sometimes called "prefix codes") (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to do this in linear time if input probabilities (also known as *weights*) are sorted.

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding, e.g., ASCII coding. Huffman coding is such a widespread method for creating prefix-free codes that the term "Huffman code" is widely used as a synonym for "prefix-free code" even when such a code is not produced by Huffman's algorithm.

Although Huffman coding is optimal for a symbol-by-symbol coding with a known input probability distribution, its optimality can sometimes accidentally be over-stated. For example, arithmetic coding and LZW coding often have better compression capability. Both these methods can combine an arbitrary number of symbols for more efficient coding, and generally adapt to the actual input statistics, the latter of which is useful when input probabilities are not precisely known or vary significantly within the stream.

5.6 HARDWARE SELECTION

There are several factors to consider prior to

Hardware selection:

1. Define system capabilities that make sense for business. Computers have proven valuable to business in the following areas:
   a. Cost reduction includes reduction of inventory, savings on space, and improved ability to predict business trends.
   b. Cost avoidance includes early detection of problems and ability to expand operations without adding clerical help.
   c. Improved service emphasizes quick availability of information to customers, improved accuracy, fast turnaround.

d. Improved profit reflects the “bottom line” of the business and its ability to keep receivables within reason.

2. Specify the magnitude of the problem; that is, clearly whether selection entails a few peripherals or a major decision concerning the mainframe.
3. Assess the competence of the in-house staff. This involves determining the expertise needed in areas such as telecommunications and database design. Acquiring a computer often results in securing temporary help for conversion. Planning for this step is extremely important.
4. Consider hardware and software as a package. This approach ensures compatibility. In fact, software should be considered first, because often the user secures the hardware and then wonders what software is available for it. Remember that software solves problems and hardware drives the software to facilitate solutions.
5. Develop a schedule for the selection process. Maintaining a schedule helps keep the project under control.
6. Provide user indoctrination. This is crucial, especially fro first-time users. Selling the system to the user staff, providing adequate training, and preparing an environment conductive to implementation are prerequisites for system acquisition.

MAJOR PHASES IN SELECTION

The selection process should be viewed as a project, and a project team should be organized with management support. In large projects the team includes one or more user representatives, an analyst, and EDP auditor, and a consultant. Several steps make up the selection process; some overlap due to the dynamic nature of selection:

1. Requirements analysis.
2. System specifications.
3. Evaluation and Validation.
4. Vendor selection.
5. Post-installation review.

5.7 SOFTWARE SELECTION

Software selection is a crucial aspect of system development. As mentioned earlier, the search starts with the software, followed by the hardware. There are two ways of acquiring software: custom-made or “off-the-shelf” packages. Today’s trend is toward purchasing packages, which represent roughly 10 percent of what are costs to develop the same in house. In addition to reduced cost, there are other advantages:

1. A good package can get the system running in a matter of days rather than the weeks or months required for “home-grown” packages.
2. MIS personnel are released for other projects.
3. Packages are generally reliable and perform according to stated documentation.
4. Minimum risks are usually associated with large-scale systems and programming efforts.
5. Delays in completing software projects in house often occur because programmers quit in midstream.
6. It is difficult to predict the cost of “home-grown” software.
7. The user has a chance of seeing how well the package performs before purchasing it.

   There are drawbacks, however, to software packages:

   1. The package may not meet user requirements adequately.
   2. Extensive modification of a package usually results in loss of the vendor’s support.
   3. The methodology for package evaluation and selection is often poorly defined. The result is a haphazard review based on a faulty process or questionable selection criteria.
   4. For first-time software package users, the overall expectation from a package is often unclear and ill defined.

It can be seen, then, that the quality of a software package cannot be determined by price alone. A systematic review is crucial.

Criteria for Software Selection

Prior to selecting the software, the project team must set up criteria fro selection. Selection criteria fall into the categories described here.

Reliability: It is the probability that the software will execute for a specified time period without a failure, weighted by the cost to the user of each failure encountered. It relates to the ease of recovery and ability to give consistent results. Fro example, a pharmacist relics on past files on patients when filling prescriptions. Information accuracy is crucial.

Functionalit: It is a definition of the facilities, performance, and other factors that the user requires in the finished product. All such information comes from the user. The following are key questions to consider:

1. Do the input transactions, files, and reports contain the necessary data elements?

2. Are all the necessary computations and processing performed according to specifications.

Capacity: Capacity refers to the capability of the software package to handle the user's requirements for size of files, number of data elements, volume of transactions should be checked.

Flexibility: It is a measures of the effort required to modify an operational program. One feature of flexibility is adaptability, which is a measure of the ease of extending the product.

Usability: This criterion refers to the effort required to operate, prepare the input, and interpret the output of a program. Additional points to be considered are portability and understandability.

Security: It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data.

Performance: It is a measure of the capacity of the software package to do what it is expected to do. This criterion focuses on throughput, or how effectively a package performs under peak loads. Each package should be evaluated for acceptance on the user's system.

Serviceability: This criterion focuses on documentation and vendor support. Complete documentation is critical for software enhancement. It includes a narrative description of the system, system logic and logic flowcharts, input-output and file descriptions and layouts, and operator instructions.

Ownership: Who owns the software once it is "sold" to the user?

Most of the standard license agreement forms essentially lease the software to the user for an indefinite time. The user does not "own" it, which means that the source code is inaccessible for modification, except by the vendor.

Minimal costs: Cost is a major consideration in deciding between in-house and vendor software. Cost-conscious users consider the following points:

1. Development and conversion costs.
2. Delivery schedule.
3. Cost and frequency of software modifications.
4. Usable life span of the package.

SELF ASSESSMENT QUESTIONS: I

ANSWER THE FOLLOWING QUESTIONS

1. Write any two features of design objectives.
2. Define Modularity.
3. Define Security.

UNIT QUESTIONS

1. Explain the features of design objectives.
2. Explain the program structure chart.
3. Describe the design structure of software.
4. Write short notes on training and conversion.
5. Explain the following
   - Hardware selection
   - Software selection

RECOMMENDETION FOR FURTHER READINGS

1. System analysis and design, Elias M.Awad.
2. System analysis and design, Lee.

ANSWERS OF SELF-ASSESSMENT QUESTIONS

1. i) To know about the design loop / process.

   ii) To be able to individually identify a problem situation.

2. Software consists of segments connected in various ways, partly due to the separate functions they perform.

3.It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data.

NOTES