System Analysis and Design

Copyright ©

This book contains the course content for System Analysis and Design.

First Edition 2013

Printed by
Universal Training Solutions Private Limited

Address
05th Floor, I-Space,
Bavdhan, Pune 411021.

Index

Contents

List of Figures

List of Tables

Abbreviations

DFD	-	Data flow diagrams
E-R model	-	Entity relationship model
GUI	-	Graphical User Interface
HIPO	-	Hierarchy plus Input Process Output
IPO	-	Input Process Output
IS	-	Information systems
RAD	-	Rapid Application Development
ROI	-	Return on Investment
SAD	-	System Analysis and Design
SDLC	-	System Development Life Cycle
SRS	-	System Requirements Specification
VTOC	-	Visual Table of Contents

Chapter I

Introduction to System Analysis and Design

Aim

The aim of this chapter is to:

- explicate the concept of system analysis
- explain the importance of system design
- elucidate the need for defining a system

Objectives

The objectives of this chapter are to:

- explicate the evolution of system life cycle
- analyse the process of system design
- enlist the benefits of system coding

Learning outcome

At the end of this chapter, you will be able to:

- define system testing
- understand the importance of coding
- identify the benefits of system design

1.1 Introduction

Systems are designed to solve problems. One can think of the systems approach as a controlled way of dealing with a problem. In this vibrant world, the subject System Analysis and Design (SAD) primarily deals with the software development activities.

Information system, basically, has three key components. They are:

- Hardware
- Software
- Data

Hardware

Hardware consists of everything in the physical layer of the information. For example, hardware can include workstations, servers, networks, fibre optic cables, telecommunication equipment, handheld computers, digital capture devices, scanners, and other technology based infrastructure.

Software

Software basically refers to the programs that manage the hardware and provide the desired results or information's. Software consists of system software and application software. System software normally manages the hardware components, which can take account of a single workstation or a global network which has control over thousands of clients. Application software contains many programs that carry day-to-day business functions and provide users with the information they require.

Data

Data is basically the raw material which is transformed into useful information by information system.

1.2 Defining a System

A collection of components that work collectively to understand some objectives forms a system. On the whole there are three major components in every system, namely processing, input and output.

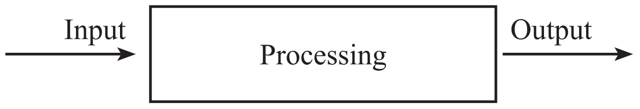


Fig.1.1 Basic system components

In a system the various components are linked with each other and they are inter-reliant. For example, human body represents a complete natural system. We are also bound by many national systems such as economic system, political system, educational system and many more. The objective of the system demands that some output is formed as a result of processing the appropriate inputs. A well-designed system also includes an additional element referred to as ‘control’ that provides a feedback to achieve desired objectives of the system.

1.3 System Life Cycle

System life cycle is an organisational procedure of developing and maintaining systems. It helps in establishing a proper system project plan, because it gives overall list of processes and sub-processes necessary for developing a system. System development life cycle means grouping of various activities. Hence 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 also means software development life cycle.

There are various phases of system development life cycle which are as follows:

- Preliminary study
- Feasibility study
- Detailed system study
- System analysis
- System design
- Coding
- Testing
- Implementation
- Maintenance

The different phases of system development life cycle are shown in figure below:

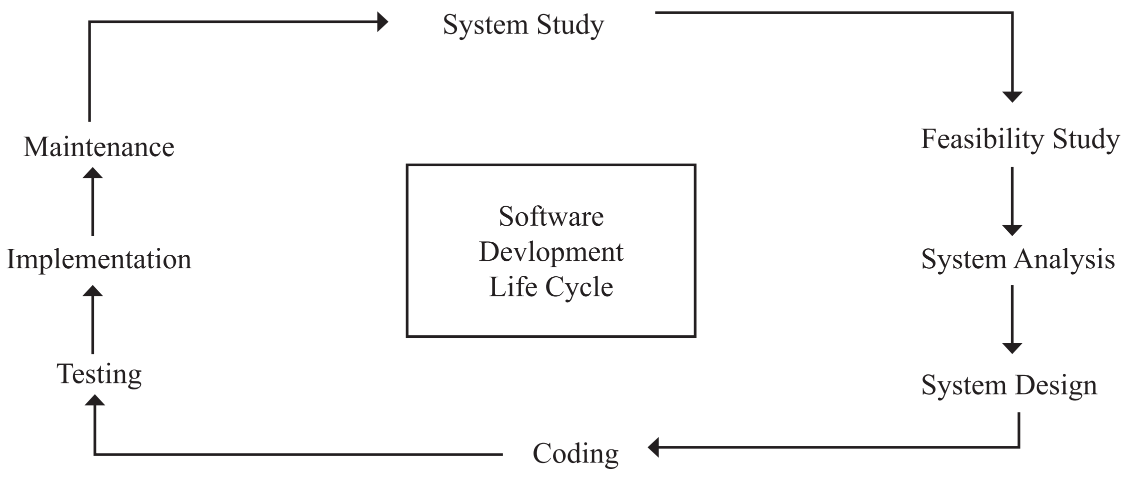


Fig.1.2 Phases of system development life cycle

1.3.1 Preliminary System Study

Preliminary system study is the primary stage of system development life cycle. This is a concise inquiry of the system under consideration and gives a clear picture of what really the physical system is? In practice, the initial system study involves the research of a 'System Proposal' which lists the Problem Definition, Terms of reference for Study, Objectives of the study, Constraints, and Expected benefits of the new system, etc. in the light of the user requirements.

The system proposal is all set by the System Analyst (who studies the system) and places it before the user management. The management may understand the proposal and the cycle proceeds to the next stage. The management may also refuse the proposal or request some modifications in the proposal. Thus, we would say that the system study phase passes through the following steps:

- Problem identification and project initiation
- Background analysis
- Inference or findings (system proposal)

1.3.2 Feasibility Study

In case the system proposal is satisfactory to the management, the next phase is to examine the practicability of the system. The practicability study is basically the test of the proposed system in the light of its workability, meeting the user's requirements, effective use of resources and of course, the cost effectiveness. These are categorised as economic, operational, technical, and schedule practicability. The main goal of practicability study is not to solve the problem but to achieve the scope. In the process of practicability study, the benefits and cost are estimated with greater accuracy to find the Return on Investment (ROI). This also defines the resources needed to complete the detailed investigation. The result is a feasibility report submitted to the management. This may be accepted or accepted with modifications or rejected. The system cycle proceeds only if the management accepts it.

1.3.3 Detailed System Study

The detailed examination of the system is carried out in accordance with the objectives of the proposed system. This involves complete study of various operations performed by a system and their relationships within and outside the system. During this entire process, 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 detailed system study. It becomes easy to draw the exact boundary of the new system under consideration using the following steps:

- Keeping in observation, the problems and new necessities
- Working out the pros and cons as well as new areas of the system

All the data and the answer must be documented in the form of detailed data flow diagrams (DFDs), data dictionary, miniature specifications and logical data structures. The main points to be discussed in this stage are:

- Requirement of what the new system is to accomplish based on the user requirements.
- Functional hierarchy presenting the functions to be performed by the new system and their connection with each other.
- Functional network, which are identical to function hierarchy but they focus on the 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).

1.3.4 System Analysis

Systems analysis is a process of collecting factual data; understand the processes involved, identifying problems and recommending feasible suggestions for improving the system performance. This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organisational goals. System Analysis also includes subdividing of complex process involving the entire system, identification of data store and manual processes.

The major objectives of systems analysis are to find answers for each business process: What is being done? How is it being done? Who is doing it? When is he doing it? Why is it being done? How can it be improved? It is more of a thinking process and involves the creative skills of the System Analyst. It attempts to give birth to a new efficient system that satisfies the current needs of the user and has scope for future growth within the organisational constraints. The result of this process is a logical system design. Systems analysis is an iterative process that continues until a preferred and acceptable solution emerges.

1.3.5 System Design

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design. Normally, the design proceeds in two stages:

- Preliminary or General Design
- Structured 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.

Structured 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-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail.

In the design stage, the programming language and the hardware and software platform in which the new system will run are also decided. There are several tools and techniques used for describing the system design of the system.

The tools and techniques are as follows:

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

Each of the above given tools are even used for designing.

The system design involves:

- Defining precisely the required system output
- Determining the data requirement for producing the output
- Determining the medium and format of files and databases
- Devising processing methods and use of software to produce output
- Determine the methods of data capture and data input
- Designing Input forms
- Designing Codification Schemes
- Detailed manual procedures
- Documenting the Design

1.3.6 Coding

The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. 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 changes, if required.

1.3.7 Testing

Before actually implementing the new system into operation, a test run of the system is done for removing 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. Sometimes, system testing is considered a part of implementation process.

Using the test data, the following test run are carried out:

- Program test
- System test

Program test
When the programs have been coded, 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 program test for each of the programs of the system and errors 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 analysed. During the result analysis, it may be found that the outputs are not matching the expected output 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.

1.3.8 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. The major steps involved in this phase are:

- Acquisition and Installation of Hardware and Software
- Conversion
- User Training
- Documentation

The hardware and the relevant software required for running the system must be made fully operational before implementation. The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system.

The database needs to be setup with security and recovery procedures fully defined. During this phase, all the programs of the system are loaded onto the user’s computer. After loading the system, training of the user starts. The 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 computerised system, working has to shift from manual to computerised working. The process is called ‘Changeover’.

The following strategies are followed for changeover of the system.

Direct Changeover
This is the complete replacement of the old system by the new system. It is a risky approach and requires comprehensive system testing and training.

Parallel run
In parallel run both the systems, i.e., computerised and manual are executed simultaneously for a certain defined period. The same data is processed by both the systems. This strategy is less risky but more expensive because of the following:

- Manual results can be compared with the results of the computerised system.
- The operational work is doubled.
- Failure of the computerised system at the early stage does not affect the working of the organisation, because the manual system continues to work, as it used to do.

Pilot run
In this type of run, the new system is run with the data from one or more of the previous periods for the whole or part of the system. The results are compared with the old system results. It is less expensive and risky than parallel run approach. This strategy builds the confidence and the errors are traced easily without affecting the operations.

The documentation of the system is also one of the most important activities in the system development life cycle. This ensures the continuity of the system. There are generally two types of documentation prepared for any system. They are:

- User or Operator Documentation
- System Documentation

The user documentation is a complete description of the system from the user's point of view detailing how to use or operate the system. It also includes the major error messages likely to be encountered by the users. The system documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description etc. This helps to understand the system and permit changes to be made in the existing system to satisfy new user needs.

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 environments. It has been seen that there are always some errors found in the systems 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.

Summary

- Systems are designed to solve problems. One can think of the systems approach as a controlled way of dealing with a problem.
- Hardware consists of everything in the physical layer of the information.
- Software basically refers to the programs that manage the hardware and provide the desired results or information's. Software consists of system software and application software.
- Data is basically the raw material which is transformed into useful information by information system.
- The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language.
- The output of the test run should match the expected results. Sometimes, system testing is considered a part of implementation process.
- The documentation of the system is also one of the most important activities in the system development life cycle.
- Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem.
- The system documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description, etc.

References

- John, W. S., Robert, B. J., Stephen, D. B., 2011. *Systems Analysis and Design in a Changing World*, Cengage Learning.
- Penny, A. K., 1992. *Introduction to systems analysis and design: a structured approach*, Wm. C. Brown Publishers.
- *An Introduction to Structured System Analysis and Design Method* [Pdf] Available at: < http://www.shere2000.co.uk/pdf/ssadm01.pdf> [Accessed 25 June 2013].
- *System Analysis* [Pdf] Available at: < http://newton.uor.edu/courses/sysanades/pdf/anaintro.pdf> [Accessed 25 June 2013].
- *Modern System Analysis and Design* [Video online] Available at: <http://www.youtube.com/watch?v=fxzPolwXGL8> [Accessed 25 June 2013].
- *Computer Technologies* [Video online] Available at: <http://www.youtube.com/watch?v=X6tPgXPNR3c> [Accessed 25 June 2013].

Recommended Reading

- Ned, K., 2006. *Systems Analysis & Design Fundamentals: A Business Process Redesign Approach,* SAGE.
- Gupta, P., 2008. *System Analysis and Design*, Firewall Media.
- Jeffrey, L. W., Lonnie, D. B., 2008. *Introduction to Systems Analysis and Design,* McGraw Hill Irwin.

Self Assessment

1. System study is the _____________ stage of system development life cycle.
    a. first
    b. second
    c. fifth
    d. sixth

2. Analysis involves a ____________ study of the current system.
    a. system
    b. data
    c. detailed
    d. software

3. _____________ is a blue print of a computer system.
    a. logical structure design
    b. software
    c. data
    d. structure design

4. In ___________ run the new system installed in parts.
    a. data
    b. pilot
    c. software
    d. components

5. In parallel run computerised and ____________ systems are executed in parallel.
    a. manual
    b. automatic
    c. software
    d. hardware

6. Which of the following statements is true?
    a. A collection of components that work together to realise some objectives forms a system.
    b. System life cycle is not an organisational process of developing and maintaining a system.
    c. System analysis and system design are the same phase of system development life cycle.
    d. Coding is not a step in system development life cycle.

7. Which of the following statements is false?
    a. A collection of components that work together to realise some objectives forms a system.
    b. Hardware consists of everything in the physical layer of the information.
    c. Data is basically the raw material which is transformed into useful information by information system.
    d. Coding is not a step in system development life cycle.

8. ___________ is basically the raw material which is transformed into useful information by information system.
    a. Information
    b. Data
    c. Software
    d. Hardware

9. ___________ consists of everything in the physical layer of the information.
    a. Hardware
    b. Software
    c. Data
    d. Components

10. The ___________documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description, etc.
    a. data
    b. software
    c. system
    d. components

Chapter II

Process of System Development

Aim

The aim of this chapter is to:

- explicate the concept of system development
- explain the importance of system development life cycle
- elucidate the need for system security

Objectives

The objectives of this chapter are to:

- explicate the evolution of waterfall model
- analyse the process of iterative development
- enlist the benefits of prototyping

Learning outcome

At the end of this chapter, you will be able to:

- define exploratory model
- understand the importance of system development
- identify the need of ad-hoc development

2.1 Introduction

Information Systems Analysis and Design is a complex and stimulating process that is used to develop and maintain computer based information systems. The analysis and design of information systems are driven from an organisational point of view. An organisation might consist of whole enterprise, specific departments or individual work groups. Information Systems Analysis and Design is, therefore, an organisational improvement process. Systems are built and rebuilt (enhanced) for organisational benefits. Benefits result by adding value during the process of creating, producing and supporting the organisation's services and products. Thus, Information Systems Analysis and Design is based on the understanding of objectives, structure and processes of organisation and the knowledge about the application of Information Technology for this purpose.

2.2 System Development Life Cycle

Most organisations find it beneficial to use standard sets of steps, called a systems development methodology, to develop and support their information systems (IS). Like many processes, the development of Information Systems often follows a life cycle called Systems Development Life Cycle. For example, a product follows a life cycle when it is created, tested and introduced in the market. Its sale increases and goes to peak point and after that it declines and a new product or next version of the existing product is introduced in the market to replace it. SDLC is a common methodology for systems development in many organisations, consisting of various phases that mark the progress of system analysis and design.

Although any life cycle appears at first glance to be a sequentially ordered set of phases but actually it is not. The specific steps and their sequence are meant to be adapted as required for a project, consistent with management approach. For example, in any given SDLC phase, the project can return to an earlier phase, if necessary. If a commercial product does not perform well just after its introduction, it may be temporarily removed from the market and improved before being re-introduced. In the system development life cycle, it is also possible to complete some activities in one phase in parallel with some other activities of another phase. Sometimes, life cycle is iterative; that is, phases are repeated as required until a satisfactory and acceptable system is found. Such an iterative approach is special characteristic of rapid application development methods, such as prototyping. Some people consider life cycle to be spiral, in which we constantly cycle through the phases at different levels of detail.

The life cycle can also be thought of a circular process in which the end of the useful life of one system leads to the beginning of another project that will develop a new version or replace an existing system altogether. However, the system development life cycle used in an organisation is an orderly set of activities conducted and planned for each development project. The skills of a system analyst are required to be applied to the entire life cycle.

2.3 Typical Tasks in the Development Process Life Cycle

Professional system developers and the customers they serve share a common goal of building information systems that effectively support business process objectives. In order to ensure that cost-effective, quality systems are developed which address an organisation's business needs, developers employ some kind of system development Process Model to direct the project's life cycle. Typical activities that are performed include the following:

- System conceptualisation
- System requirements and benefits analysis
- Project adoption and project scoping
- System design
- Specification of software requirements
- Architectural design
- Detailed design
- Unit development
- Software integration & testing
- System integration & testing

- Installation at site
- Site testing and acceptance
- Training and documentation
- Implementation
- Maintenance

2.4 Process Model/Life-Cycle Variations

While nearly all system development efforts engage in some combination of the above tasks, they can be differentiated by the feedback and control methods employed during development and the timing of activities. Most system development Process Models in use today have evolved from three primary approaches: Ad-hoc Development, Waterfall Model, and the Iterative process.

2.5 Ad-hoc Development

Early systems development often took place in a rather chaotic and haphazard manner, relying entirely on the skills and experience of the individual staff members performing the work. Today, many organisations still practice Ad-hoc Development either entirely or for a certain subset of their development (e.g. small projects).

The Software Engineering Institute at Carnegie Mellon University points out that with Ad-hoc Process Models, "process capability is unpredictable because the software process is constantly changed or modified as the work progresses. Schedules, budgets, functionality, and productquality are generally (inconsistent). Performance depends on the capabilities of individuals and varies with their innate skills, knowledge, and motivations. There are few stable software processes in evidence, and performance can be predicted only by individual rather than organisational capability."

"Even in undisciplined organisations, however, some individual software projects produce excellent results. When such projects succeed, it is generally through the heroic efforts of a dedicated team, rather than through repeating the proven methods of an organisation with a mature software process.In the absence of an organisation-wide software process, repeating results depends entirely on having the same individuals available for the next project. Success thatrests solely on the availability of specific individuals provides no basis for long-term productivity and quality improvement throughout an organisation."

Fig.2.1 Ad-hoc development

(Source: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf)

2.6 The Waterfall Model

The Waterfall Model is the earliest method of structured system development. Although it has come under attack in recent years for being too rigid and unrealistic when it comes to quickly meeting customer's needs, the Waterfall Model is still widely used. It is attributed with providing the theoretical basis for other Process Models, because it most closely resembles a "generic" model for software development.

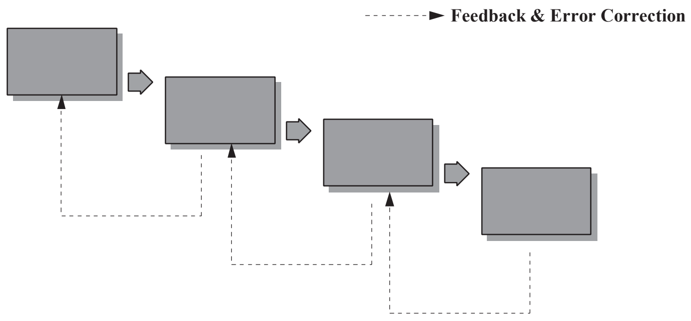


Fig.2.2 Waterfall Model
(Source: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf)

The Waterfall Model consists of the following steps:

System Conceptualisation

System Conceptualisation refers to the consideration of allaspects of the targeted business function or process, with the goals of determining how each of those aspects relates with one another, and which aspects will be incorporated into the system.

Systems Analysis

This step refers to the gathering of system requirements, with the goalof determining how these requirements will be accommodated in the system. Extensive communication between the customer and the developer is essential.

System Design

Once the requirements have been collected and analyzed, it is necessaryto identify in detail how the system will be constructed to perform necessary tasks. More specifically, the System Design phase is focused on the data requirements (What information will be processed in the system?), the software construction (How will the application be constructed?), and the interface construction (What will the system look like? What standards will be followed?).

Coding

Also known as programming, this step involves the creation of the systemsoftware. Requirements and systems specifications from the System Design step are translated into machine readable computer code.

Testing

As the software is created and added to the developing system, testing isperformed to ensure that it is working correctly and efficiently. Testing is generallyfocused on two areas: internal efficiency and external effectiveness. The goal of externaleffectiveness testing is to verify that the software is functioning according to system design, and that it is performing all necessary functions or sub-functions. The goal of internal testing is to make sure that the computer code is efficient, standardised, and welldocumented. Testing can be a labour-intensive process, due to its iterative nature.

2.6.1 Challenges Associated with the Waterfall Model

Although the Waterfall Model has been used extensively over the years in the production of many quality systems, it is not without its problems. In recent years it has come under attack, due to its rigid design and inflexible procedure. The criticisms fall into the following categories:

- Real projects rarely follow the sequential flow that the model proposes.
- At the beginning of most projects there is often a great deal of uncertainty about requirements and goals, and it is therefore difficult for customers to identify these criteria on a detailed level. The model does not accommodate this natural uncertainty very well.
- Developing a system using the Waterfall Model can be a long, painstaking process that does not yield a working version of the system until late in the process.

2.7 Iterative Development

The problems with the Waterfall Model created a demand for a new method of developing systems which could provide faster results, require less up-front information, and offer greater flexibility. With Iterative Development, the project is divided into small parts. This allows the development team to demonstrate results earlier on in the process and obtain valuable feedback from system users. In a variation of this model, the software products which are produced at the end of each step (or series of steps) can go into production immediately as incremental releases.

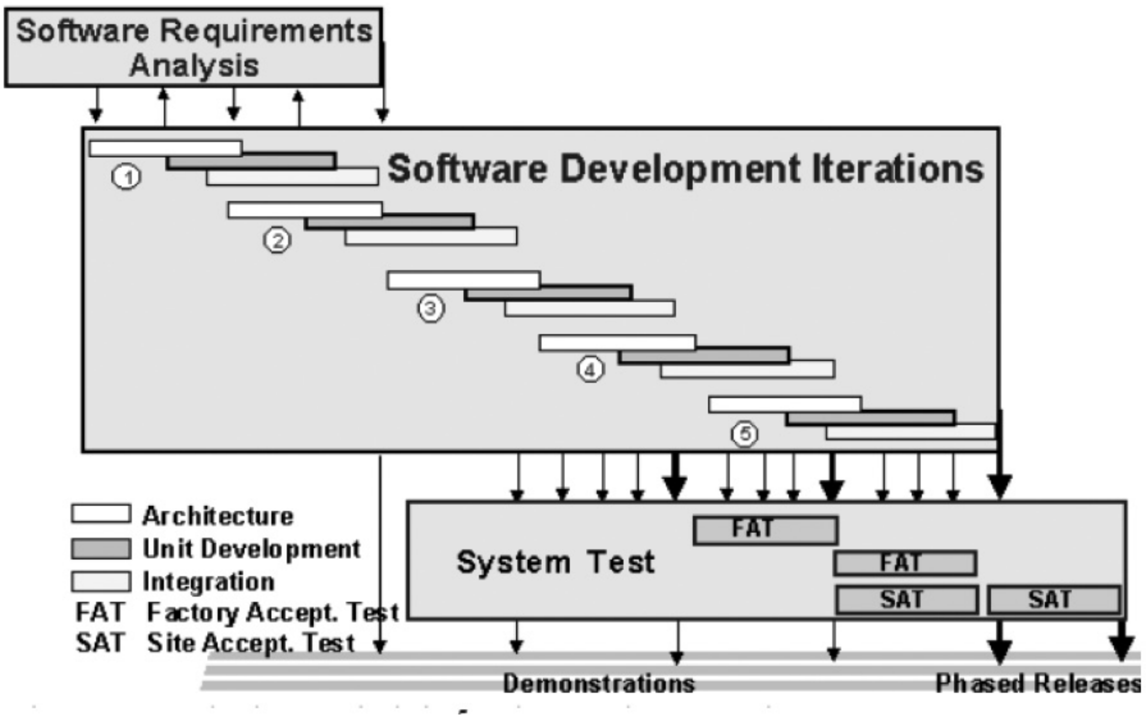


Fig.2.3 Iterative Development
(Source: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf)

2.7.1 Challenges Associated with the Iterative Model

While the Iterative Model addresses many of the problems associated with the Waterfall Model, it does present new challenges. They are as follows:

- The user community needs to be actively involved throughout the project. While this involvement is a positive for the project, it is demanding on the time of the staff and can add project delay.
- Communication and coordination skills take centre stage in project development.
- Informal requests for improvement after each phase may lead to confusion;a controlled mechanism for handling substantive requests needs to be developed.
- The Iterative Model can lead to “scope creep,” since user feedback following each phase may lead to increased customer demands. As users see the system develop, they may realise the potential of other system capabilities which would enhance their work.

2.7.2 Variations on Iterative Development

A number of Process Models have evolved from the Iterative approach. All of these methods produce some demonstrable software product early on in the process in order to obtain valuable feedback from system users or other members of the project team.

2.8 Prototyping

The Prototyping Model was developed on the assumption that it is often difficult to know all ofyour requirements at the beginning of a project. Typically, users know many of the objectives that they wish to address with a system, but they do not know all the nuances of the data, nor do they know the details of the system features and capabilities. The Prototyping Model allows for these conditions, and offers a development approach that yields results without first requiring all information up-front.

When using the Prototyping Model, the developer builds a simplified version of the proposedsystem and presents it to the customer for consideration as part of the development process. The customer in turn provides feedback to the developer, who goes back to refine the system requirements to incorporate the additional information. Often, the prototype code is thrown away and entirely new programs are developed once requirements are identified.

There are a few different approaches that may be followed when using the Prototyping Model:

- Creation of the major user interfaces without any substantive coding in the background in order to give the users a “feel” for what the system will look like,
- Development of an abbreviated version of the system that performs a limited subset of functions; development of a paper system (depicting proposed screens, reports, relationships etc.), or
- Use of an existing system or system components to demonstrate some functions that will be included in the developed system.

2.8.1 Steps in Prototyping

There are various steps involved in Prototyping. They are as follows:

Requirements Definition/Collection

It is similar to the conceptualisation phase of theWaterfall Model, but not as comprehensive. The information collected is usually limited to a subset of the complete system requirements.

Design

Once the initial layer of requirements information is collected, or newinformation is gathered, it is rapidly integrated into a new or existing design so that it may be folded into the prototype.

Prototype Creation/Modification

The information from the design is rapidly rolled intoa prototype. This may mean the creation/modification of paper information, new coding, or modifications to existing coding.

Assessment

The prototype is presented to the customer for review. Comments andsuggestions are collected from the customer.

Prototype refinement

Information collected from the customer is digested and theprototype is refined. The developer revises the prototype to make it more effective and efficient.

System implementation

In most cases, the system is rewritten once requirements areunderstood. Sometimes, the Iterative process eventually produces a working system that can be the cornerstone for the fully functional system.

2.8.2 Challenges Associated with the Prototyping Model

The criticisms of the Prototyping Model generally fall into the following categories:

Prototyping can lead to false expectations

Prototyping often creates a situation wherethe customer mistakenly believes that the system is "finished" when in fact it is not. More specifically, when using the Prototyping Model, the pre-implementation versions of a system are really nothing more than one-dimensional structures. The necessary, behind the-scenes work such as database normalisation, documentation, testing, and reviews for efficiency have not been done. Thus, the necessary underpinnings for the system are not in place.

Prototyping can lead to poorly designed systems.

As the primary goal ofPrototyping is rapid development, the design of the system can sometimes suffer because the system is built in a series of "layers" without a global consideration of the integration of all other components. While initial software development is often built to be a "throwaway," attempting to retroactively produce a solid system design can sometimes be problematic.

2.8.3 Variation of the Prototyping Model

A popular variation of the Prototyping Model is called Rapid Application Development (RAD). RAD introduces strict time limits on each development phase and relies heavily on rapid application tools which allow for quick development.

2.9 The Exploratory Model

In some situations it is very difficult, if not impossible, to identify any of the requirements for a system at the beginning of the project. Theoretical areas such as Artificial Intelligence are candidates for using the Exploratory Model because much of the research in these areas is basedon guess-work, estimation, and hypothesis. In these cases an assumption is made as to how the system might work and then rapid iterations are used to quickly incorporate suggested changes and build a usable system. A distinguishing characteristic of the Exploratory Model is the absence of precise specifications. Validation is based on adequacy of the end result and not on its adherence to pre-conceived requirements.

The Exploratory Model is extremely simple in its construction.It is composed of the following steps:

- Initial Specification Development: Using whatever information is immediately available, a brief System Specification is created to provide a rudimentary starting point.
- System Construction/Modification: A system is created and/or modified according to whatever information is available.
- System Test: The system is tested to see what it does, what can be learned from it, and how it may be improved.
- System Implementation: After much iteration of the previous two steps produce satisfactory results, the system is dubbed as "finished" and implemented.

2.9.1 Challenges Associated with the Exploratory Model

There are numerous criticisms of the Exploratory Model. They are:

- It is limited to use with very high-level languages that allow for rapid development, suchas LISP.
- It is difficult to measure or predict its cost-effectiveness.
- As with the Prototyping Model, the use of the Exploratory Model often yields inefficientor crudely designed systems, since no forethought is given as to how to produce a streamlined system.

2.10 The Spiral Model

The Spiral Model was designed to include the best features from the Waterfall and Prototyping Models, and introduces a new component - risk-assessment. The term “spiral” is used to describe the process that is followed as the development of the system takes place. Similar to the Prototyping Model, an initial version of the system is developed, and then repetitively modified based on input received from customer evaluations. Unlike the Prototyping Model, however, the development of each version of the system is carefully designed using the steps involved in theWaterfall Model. With each iteration around the spiral (beginning at the centre and working outward), progressively more complete versions of the system are built.

Risk assessment is included as a step in the development process as a means of evaluating each version of the system to determine whether or not development should continue. If the customer decides that any identified risks are too great, the project may be halted. For example, if a substantial increase in cost or project completion time is identified during one phase of risk assessment, the customer or the developer may decide that it does not make sense to continue with the project, since the increased cost or lengthened timeframe may make continuation of the project impractical or unfeasible.

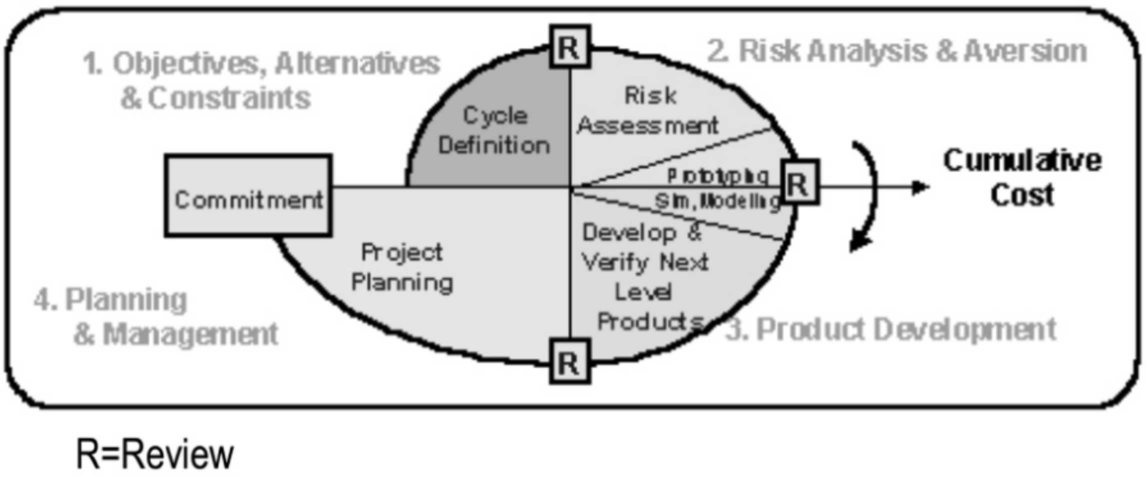

1. Objectives, Alternatives & Constraints
2. Risk Analysis & Aversion
Cycle Definition
Risk Assessment
Prototyping Sim, Modelling
Commitment
Cumulative Cost
Project Planning
Develop & Verify Next Level Products
4. Planning & Management
3. Product Development
R
R
R
R=Review

Fig. 2.4 Spiral Model
(Source: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf)

The Spiral Model is made up of the following steps:
Project objectives
It is similar to the system conception phase of the Waterfall Model.Its objectives are determined, possible obstacles are identified and alternative approaches are weighed.

Risk assessment
The possible alternatives are examined by the developer, and associatedrisks/problems are identified. The resolutions of the risks are evaluated and weighed in the consideration of project continuation. Sometimes prototyping is used to clarify needs.

Engineering and production
Detailed requirements are determined and the softwarepiece is developed.

Planning and management
The customer is given an opportunity to analyze the resultof the version created in the Engineering step and to offer feedback to the developer.

2.10.1 Challenges Associated with the Spiral Model

Due to the relative newness of the Spiral Model, it is difficult to assess its strengths and weaknesses. However, the risk assessment component of the Spiral Model provides both developers and customers with a measuring tool that earlier Process Models do not have. Themeasurement of risk is a feature that occurs every day in real-life situations, but (unfortunately)not as often in the system development industry. The practical nature of this tool helps to make the Spiral Model a more realistic Process Model than some of its predecessors.

2.11 Rescue Model

The basic premise behind the Reuse Model is that systems should be built using existingcomponents, as opposed to custom-building new components. The Reuse Model is clearly suitedto Object-Oriented computing environments, which have become one of the premieretechnologies in today’s system development industry.Within the Reuse Model, libraries of software modules are maintained that can be copied for usein any system. These components are of two types: procedural modules and database modules.

When building a new system, the developer will “borrow” a copy of a module from the systemlibrary and then plug it into a function or procedure. If the needed module is not available, thedeveloper will build it, and store a copy in the system library for future usage. If the modules arewell engineered, the developer with minimal changes can implement them.

The Reuse Model consists of the following steps:
Definition of requirements
Initial system requirements are collected. Theserequirements are usually a subset of complete system requirements.

Definition of objects
The objects, which can support the necessary system components,are identified.

Collection of objects
The system libraries are scanned to determine whether or not theneeded objects are available. Copies of the needed objects are downloaded from the system.

Creation of customised objects
Objects that have been identified as needed, but thatare not available in the library are created.

Prototype assembly
A prototype version of the system is created and/or modified usingthe necessary objects.

Prototype evaluation
The prototype is evaluated to determine if it adequately addressescustomer needs and requirements.

Refinedrequirements
Requirements are further refined as a more detailed versionof the prototype is created.

Objects refinement
Objects are refined to reflect the changes in the requirements.

2.11.1 Challenges Associated with the Reuse Model

A general criticism of the Reuse Model is that it is limited for use in object-oriented development environments. Although this environment is rapidly growing in popularity, it is currently used in only a minority of system development applications.

2.12 Creating and Combining Models

In many cases, parts and procedures from various Process Models are integrated to support system development. This occurs because most models were designed to provide a framework for achieving success only under a certain set of circumstances. When the circumstances change beyond the limits of the model, the results from using it are no longer predictable. When this situation occurs it is sometimes necessary to alter the existing model to accommodate the change in circumstances, or adopt or combine different models to accommodate the new circumstances.

The selection of an appropriate Process Model hinges primarily on two factors: organisational environment and the nature of the application. Frank Land, from the London School of Economics, suggests that suitable approaches to system analysis, design, development, and implementation be based on the relationship between the information system and its organisationalenvironment.

The four categories of relationships identified are as follows:
The Unchanging Environment
Information requirements are unchanging for thelifetime of the system (e.g. those depending on scientific algorithms). The requirements can be stated unambiguously and comprehensively. A high degree of accuracy is essential. In this environment, formal methods (such as the Waterfall or Spiral Models) would provide the completeness and precision required by the system.

The Turbulent Environment
The organisation is undergoing constant change andsystem requirements are always changing. A system developed on the basis of the conventional Waterfall Model would be, in part; already obsolete by the time it is implemented. Many business systems fall into this category. Successful methods wouldinclude those, which incorporate rapid development, some throwaway code (such as in Prototyping), the maximum use of reusable code, and a highly modular design.

The Uncertain Environment
The requirements of the system are unknown oruncertain. It is not possible to define requirements accurately ahead of time because the situation is new or the system being employed is highly innovative. Here, the development methods must emphasise learning. Experimental Process Models, which take advantage of prototyping and rapid development, are most appropriate.

The Adaptive Environment
The environment may change in reaction to the systembeing developed, thus initiating a changed set of requirements. Teaching systems and expert systems fall into this category. For these systems, adaptation is the key and the methodology must allow for a straightforward introduction of new rules.

Summary

- Information Systems Analysis and Design is a complex and stimulating process that is used to develop and maintain computer based information systems.
- Most organisations find it beneficial to use standard sets of steps, called a systems development methodology, to develop and support their information systems (IS).
- In the system development life cycle, it is also possible to complete some activities in one phase in parallel with some other activities of another phase.
- Professional system developers and the customers they serve share a common goal of building information systems that effectively support business process objectives.
- The Reuse Model is clearly suitedto Object-Oriented computing environments, which have become one of the premieretechnologies in today's system development industry.
- The selection of an appropriate Process Model hinges primarily on two factors: organisational environment and the nature of the application.
- The risk assessment component of the Spiral Model provides both developers and customers with a measuring tool that earlier Process Models do not have.
- The term "spiral" is used to describe the process that is followed as the development of the system takes place.
- The customer is given an opportunity to analyse the result of the version created in the Engineering step and to offer feedback to the developer.
- Risk assessment is included as a step in the development process as a means of evaluating each version of the system to determine whether or not development should continue.
- Information requirements are unchanging for thelifetime of the system (e.g. those depending on scientific algorithms). Requirements can be stated unambiguously and comprehensively.
- The Prototyping Model was developed on the assumption that it is often difficult to know all ofyour requirements at the beginning of a project.
- When using the Prototyping Model, the developer builds a simplified version of the proposedsystem and presents it to the customer for consideration as part of the development process.

References

- Bellgran, M., Sèafsten, K., 2010.*Production Development: Design and Operation of Production Systems*, Springer.
- Meister, D., Thomas P. E., 2001.*Human Factors in System Design, Development, and Testing*, Psychology Press.
- *Systems Development Life-Cycle Policy* [Pdf] Available at: <http://www.house.gov/content/cao/procurement/ref-docs/SDLCPOL.pdf> [Accessed 25 June 2013].
- *System Development Lifecycle* [Pdf] Available at: <http://www.security.mtu.edu/policies-procedures/SystemDevelopmentLifecycle.pdf> [Accessed 25 June 2013].
- *MIS- Business Process and Information System.mp4* [Video online] Available at: <http://www.youtube.com/watch?v=ikNlsqD0bME> [Accessed 25 June 2013].
- *Software Development Life Cycle [SDLC]*[Video online] Available at: <http://www.youtube.com/watch?v=xtpyjPrpyX8> [Accessed 4 July 2013].

Recommended Reading

- Naveen, P., Colette, R., Barbara, P., 1993.*Information system development process: proceedings of the IFIP WG8.1 Working Conference on Information Development Process*, North-Holland.
- Richard, W. P., Anne, S.M., 2007.*Human-System Integration in the System Development Process: A New Look*, National Academies Press.
- Jeffrey, O. G., 2010.*System Requirements Analysis*, Academic Press.

Self Assessment

1. The ___________ was developed on the assumption that it is often difficult to know all ofyour requirements at the beginning of a project.
   a. Spiral Model
   b. Prototyping Model
   c. Reuse Model
   d. Information system

2. The ____________ was designed to include the best features from the Waterfall and Prototyping Models, and introduces a new component- risk-assessment.
   a. Spiral Model
   b. Prototyping Model
   c. Reuse Model
   d. Data Model

3. A popular variation of the Prototyping Model is called _____________.
   a. Rare Application Development
   b. Rapid Advanced Development
   c. Rapid Application Development
   d. Rare Advanced Delay

4. The_________, which can support the necessary system components, are identified.
   a. data
   b. design
   c. information
   d. objects

5. The _________ is evaluated to determine if it adequately addressescustomer needs and requirements.
   a. prototype
   b. object
   c. data
   d. design

6. An/A________ might consist of whole enterprise, specific departments or individual work groups.
   a. team
   b. group
   c. organisation
   d. management

7. Which of the following statements is true?
   a. Information Systems Analysis and Design is a complex and stimulating process that is used to develop and maintain computer based information systems.
   b. The Reuse Model was developed on the assumption that it is often difficult to know all ofyour requirements at the beginning of a project.
   c. A popular variation of the Prototyping Model is called Rapid Advanced Development.
   d. The object is evaluated to determine if it adequately addressescustomer needs and requirements.
8. Which of the following statements is false?

a. The Prototyping Model is clearly suitedto Object-Oriented computing environments, which have become one of the premieretechnologies in today’s system development industry.
b. Information Systems Analysis and Design is a complex and stimulating process that is used to develop and maintain computer based information systems.
c. The organisation is undergoing constant change andsystem requirements are always changing.
d. A prototype version of the system is created and/or modified using the necessary objects.

9. A general criticism of the ___________ is that it is limited for use in object-oriented development environments.
   a. Prototyping Model
   b. Reuse Model
   c. Data model
   d. Object Model

10. Requirements and systems specifications from the ____________ step are translated into machine readable computer code.
    a. System Design
    b. Data Design
    c. Information Design
    d. Object Design

Chapter III

Introduction to Documentation of Systems

Aim

The aim of this chapter is to:

- introducedocumentation
- explain the importance of documentation
- elucidate the need for system requirement specifications

Objectives

The objectives of this chapter are to:

- explicate the evolution of documentation
- analyse the process of documentation
- enlist the tools specific for system design

Learning outcome

At the end of this chapter, you will be able to:

- define system design
- understand the importance of documentation
- identify the various timing diagrams

3.1 Introduction

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

3.2 Concept of Documentation

Documentation may be defined as the process of communicating about the system. The person who is dependable for this communication is called documenter. It may benoted that documenter is not responsible for the accuracy of the information, and his job is just to communicate or transfer the information.

The ISO standard ISO/IEC 12207:1995 describes documentation "as a supporting activity to record information produced by a system development life cycle process."

3.2.1 Importance of Documentation

Documentation is needed because it is:

- a means for transfer of knowledge and details about description of the system
- to communicate among different teams of the software project
- to help corporate audits and other requirements of the organization
- to meet regulatory demand
- needed for IT infrastructure management and maintenance
- needed for migration to a new software platform

Document communicates the information about the system targeted at different audience. It explains the system. The ever-increasing complexity of information system requires emphasis on well-established system of documentation. Every information system should be delivered along with an accurate and understandable document to those who will use the software.

Traditionally, documentation was done after the development of the software is completed. However, as the software development process is becoming complex and involved, documentation has become an integral part of each system development process. Documentation is now carried out at every stage as a part of development process. When the process of documentation is undertaken as a separate process, it requires planning in its own right. The figure below shows, how at the development process, documentation is done alongside each step. Design and development activities of software depend on a certain base document.

Documentation is to be carried out before actually implementing the design. In such a case, any flaw in design identified can be changed in the document thereby saving cost and time during implementation.If documentation is being developed for an existing software, then documentation is done alongside the software development process.

3.3 The Process of Documentation

The various steps involved in the process of documentation are as follows:

Collection of source material

The very first step of any documentation process is to acquire the required source material for preparation of document. The material is collected including specifications, formats, screen layouts and report layouts. A copy of the operational software is helpful for preparing the documentation for user.

Documentation plan
The documenter is responsible for preparation of a documentation plan, which specifies the details of the work to be carried out to prepare the document. It also defines and the target audience.

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

Creation of document
The document is prepared with the help of document generator.

Testing of document
The document created is tested for usability as required by the target audience.

Maintaining thedocument
Once the document is created and distributed, it must be kept up to date with new version of the software product. It must be ensured that the latest document is available to the user of the software.

Figure 3.1 depicts the various stages involved in the process of documentation.

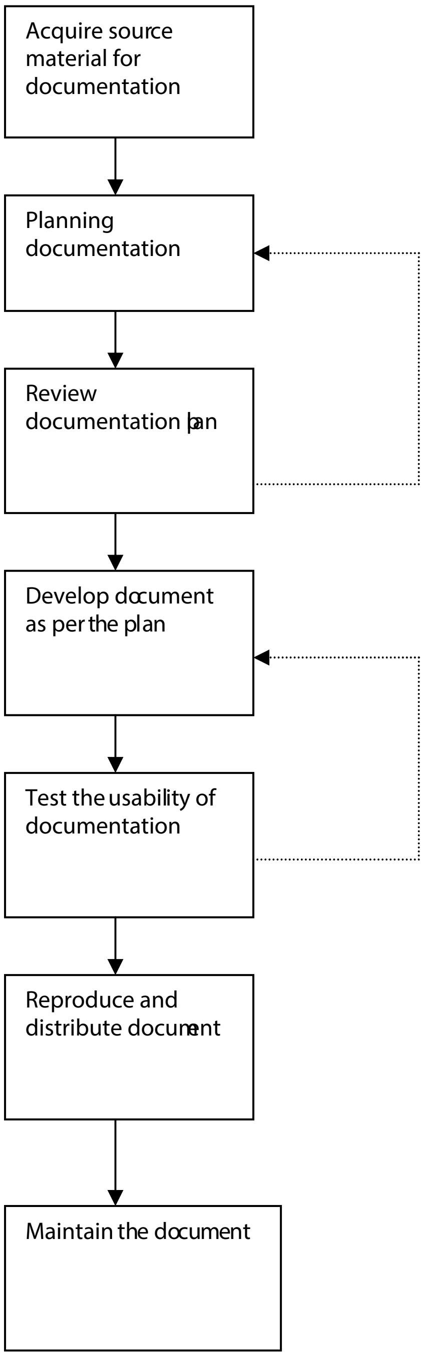

Acquire source material for documentation
Planning documentation
Review documentation plan
Develop document as per the plan
Test the usability of documentation
Reproduce and distribute document
Maintain the document

Fig.3.1. Stages in the process of documentation

3.4 Types of Documentation

Any kind of software project is associated with a large number of documents depending on the complexity of the project. Documentation that is associated with system development has a number of requirements. They are used by different types of audience in different ways which are as follows:

- They act as a means of communication between the members of development team
- Documents are used by maintenance engineer
- Documents are used by the user for operation of the software
- Documents are used by system administrator to administer the system

3.4.1 System Requirements Specification

System requirement specification is a set of complete and precisely stated properties along with the constraints of the system that the software must satisfy. A well designed software requirements specification establishes boundaries and solutions of system to develop useful software. All tasks, however minute, should not be underestimated and must form part of the documentation.

Requirements of SRS

The SRS should specify only the external system behaviour and not the internal details. It also specifies any constraints imposed on implementation. A good SRS is flexible to change and acts as a reference tool for system developer, administrator and maintainer.

Characteristics of a System Requirements Specification (SRS)

The characteristics of a SRS are as follows:

- All the requirements must be stated unambiguously. Every requirement stated has only one interpretation. Every characteristic of the final product must be described using a single and unique term.
- It should be complete. The definition should include all functions and constraints intended by the system user. In addition to the requirements of the system as specified by the user, it must conform to any standard that applies to it.
- The requirements should be realistic and achievable with current technology. There is no point in specifying requirements which are unrealisable using existing hardware and software technology. It may be acceptable to anticipate some hardware developments, but developments in software technology are much less predictable.
- It must be verifiable and consistent. The requirements should be shown to be consistent and verifiable. The requirements are verified by system tester during system testing. So, all the requirements stated must be verifiable to know conformity to the requirements. No requirement should conflict with any other requirement.
- It should be modifiable. The structure and style of the SRS are such that any necessary changes to the requirements can be made easily, completely and consistently.
- It should be traceable to other requirements and related documents. The origin of each requirement must be clear. The SRS should facilitate the referencing of each requirement for future development or enhancement of documentation. Each requirement must refer to its source in previous documents.
- SRS should not only address the explicit requirement but also implicit requirements that may come up during the maintenance phase of the software. It must be usable during operation and maintenance phase. The SRS must address the needs of the operation and maintenance phase, including the eventual replacement of the software.

Rules for Specifying Software Requirements

The rules for specifying software requirements are as follows:

- Apply and use an industry standard to ensure that standard formats are used to describe the requirements. Completeness and consistency between various documents must be ensured.
- Use standard models to specify functional relationships, data flow between the systems and sub-systems and data structure to express complete requirements.
- Limit the structure of paragraphs to a list of individual sentences to increase the tractability and modifiability of each requirement and to increase the ability to check for completeness. It helps in modifying the document when required.
- Phrase each sentence to a simple sentence. This is to increase the verifiability of each requirement stated in the document.

3.4.2 System Design Specification

The system design specification or software design specification as referred to have a primary audience, the system implementer or coder. It is also an important source of information for the system verification and testing. The system design specification gives a complete understanding of the details of each component of the system and its associated algorithms etc.

The system design specification documents all as to how the requirements of the system are to be implemented. It consists of the final steps of describing the system in detail before the coding starts.The system design specification is developed in a two stage process: In the first step, design specification generally describes the overall architecture of the system at a higher level. The second step provides the technical details of low-level design, which will guide the implementer. It describes exactly what the software must perform to meet the requirements of the system.

Tools for describing design

Various tools are used to describe the higher level and lower level aspects of system design. The following are some of the tools that can be used in the System Design Specification to describe various aspects of the design.

Data dictionary

It consists of definition of all of the data and control elements used in the software product or sub system. Complete definition of each data item and its synonyms are included in the data dictionary. A data dictionary may consist of description of data elements and definitions of tables.

- Description of data element
  - Name and aliases of data item (its formal name).
  - Uses (which processes or modules use the data item; how and when the data item is used).
  - Format (standard format for representing the data item).
  - Additional information such as default values, initial value(s), limitations and constraints that are associated with the data elements.
- Table definitions
  - Table name and Aliases.
  - Table owner or database name.
  - Key order for all the tables, possible keys including primary key and foreign key.
  - Information about indexes that exist on the table.

Database schema

Database schema is a graphical presentation of the whole database. Data retrieval is made possible by connecting various tables through keys. Schema can be viewed as a logical unit from programmer's point of view.

E-R model

Entity-relationship model is database analysis and design tool. It lists real-life application entities and defines the relationship between real life entities that are to be mapped to database. E-R model forms basis for database design.

Security model

The database security model associates users, groups of users or applications with database access rights.

Trade-off matrix

It is a matrix that is used to describe decision criteria and relative importance of each decision criterion. This allows comparison of each alternative in a quantifiable term.

Decision table

A decision table shows the way the system handles input conditions and subsequent actions on the event. A decision table is composed of rows and columns, separated into four separate quadrants.

Input Conditions	Condition Alternatives
Actions	Subsequent action Entries

Table 3.1 Decision table

3.5 Timing diagram

It describes the timing relationships among various functions and behaviors of each component. They are used to explore the behaviors of one or more objects throughout a given period of time. This diagram is specifically useful to describe logic circuits.

State machine diagram

State machine diagrams are good at exploring the detailed transitions between states as the result of events. A state machine diagram is shown in figure below.

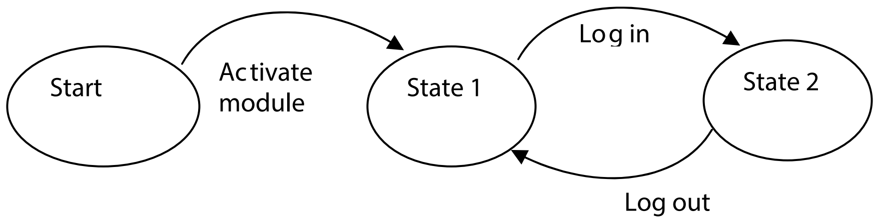


Fig.3.2 A state machine diagram

Object interaction diagram

It illustrates the interaction between various objects of an object-oriented system. Please refer to figure 3.3. This diagram consists of directed lines between clients and servers. Each box contains the name of the object. This diagram also shows conditions for messages to be sent to other objects. The vertical distance is used to show the life of an object.

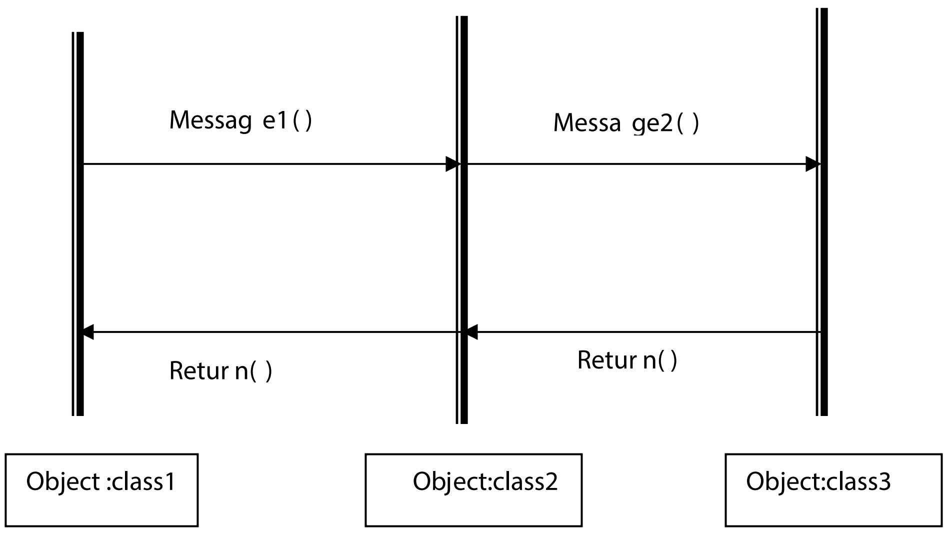


Fig.3.3 An object interaction diagram

A Flow chart shows the flow of processing control as the program executes. Please refer to figure 3.4.

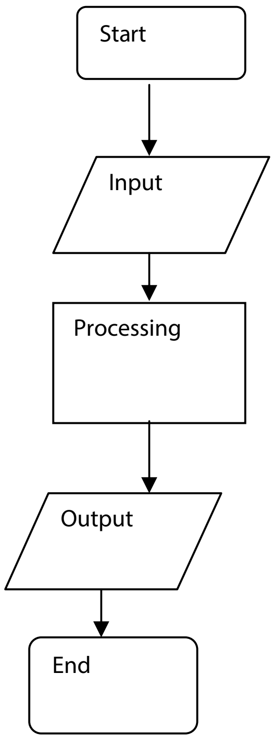


Fig. 3.4 The flow of processing control

Inheritance diagram

It is a design diagram work product that primarily documents the inheritance relationships between classes and interfaces in object-oriented modeling. The standard notation consists of one box for each class. The boxes are arranged in a hierarchical tree according to their inheritance characteristics. Each class box includes the class name, its attributes, and its operations. Figure 3.5 shows a typical inheritance diagram.

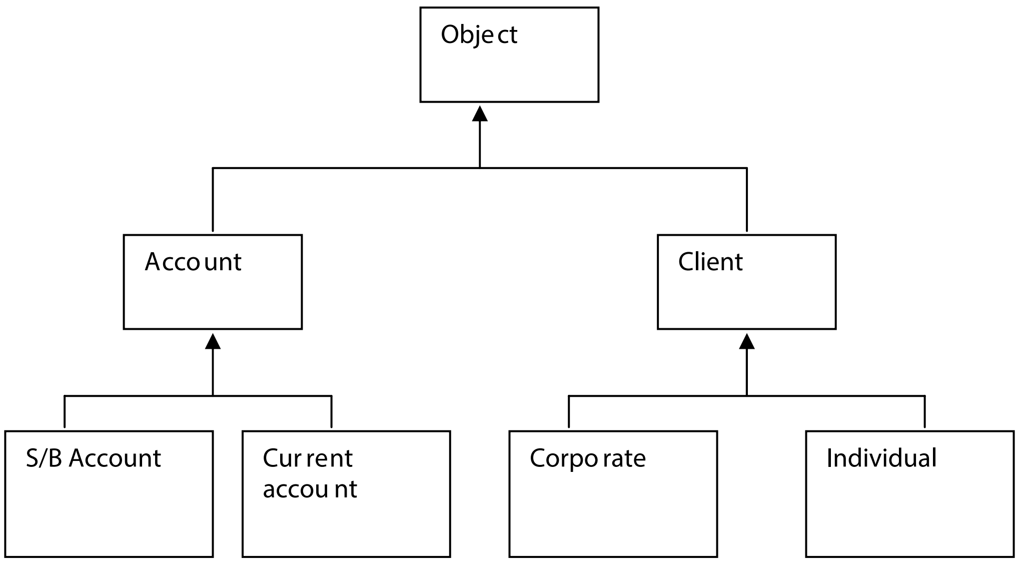


Fig. 3.5: A typical inheritance diagram

Aggregation diagram
The E-R model cannot articulate relationships among relationships. An aggregation diagram shows relationships among objects. When a class is formed as a collection of other classes, it is called an aggregation relationshipbetween these classes. Each module will be represented by its name.

The relationship will be indicated by a directed line from container to container. The directed line is labeled with the cardinality of the relationship. It describes "has a" relationship. Figure 3.6 shows an aggregation between two classes (circle has a shape).

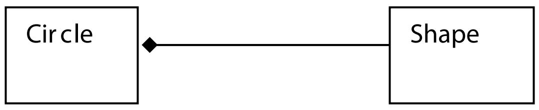


Fig.3.6 Aggregation diagram

Structure chart
A structure chart is a tree of sub-routines in a program (Refer to figure 3.7). It indicates the interconnections among the sub-routines. The sub-routines should be labelled with the same name used in the pseudo code.

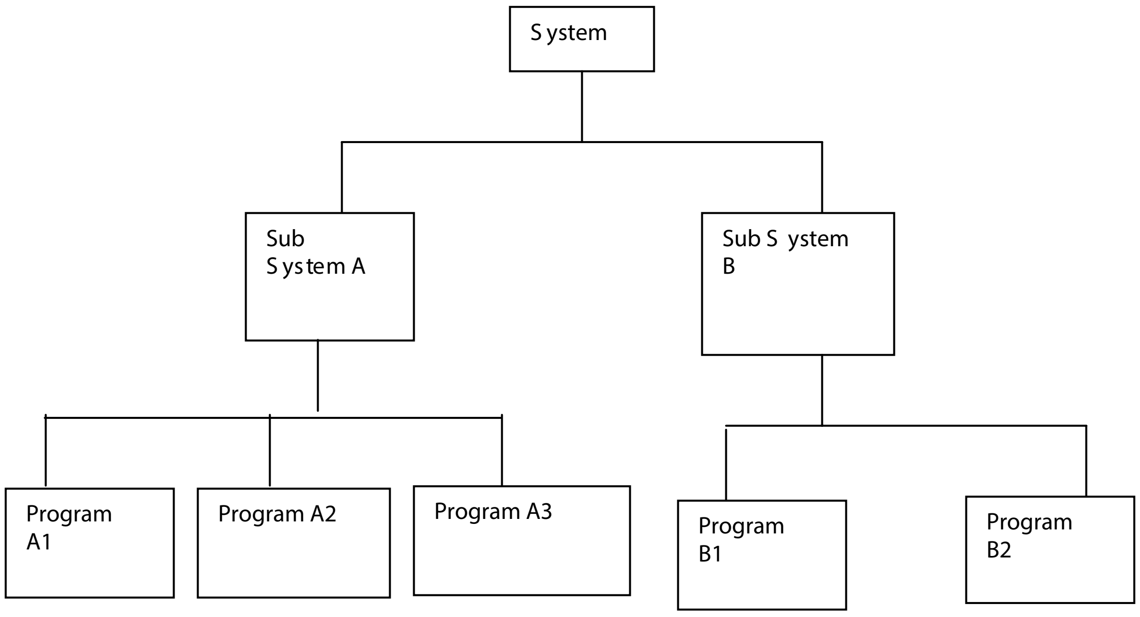


Fig. 3.7 A structure chart

Pseudocode
Pseudocode is a kind of prepared English for describing algorithms in an easily readable and modular form. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax in which the code is going to be written. The pseudocode needs to be complete. It describes the entire logic of the algorithm so that implementation becomes a routine mechanical task of translating line by line into source code of the target language. Thus, it must include flow of control.

This helps in describing how the system will solve the given problem."Pseudocode" does not refer to a precise form of expression. It refers to the simple use of Standard English. Pseudocode must use a restricted subset of English in such a way that it resembles a good high level programming language. Figure below shows an example of pseudo code.

```
        IF HoursWorked>MaxWorkHour THEN
            Display overtime message
ELSE
            Display regular time message
ENDIF
```

Fig.3.8 Example of a Pseudocode

3.5.1 Test design document

During system development, this file provides the information needed for adequate testing. It also lists approaches, procedures and standards to ensure that a quality product that meets the requirement of the user is produced. This document is generally supplemented by documents like schedules, assignments and results. A record of the final result of the testing should be kept externally.

This document provides valuable input for the maintenance phase.

The following IEEE standards describe the standard practices on software test and documentation:

- 829-1998 IEEE Standard for Software Test Documentation
- 1008-1987 (R1993) IEEE Standard for Software Unit Testing
- 1012-1998 IEEE Standard for Software Verification and Validation

The following is the typical content of Test Design Document:
Introduction

Purpose
The purpose of this document and its intended audience are clearly stated.

Scope
Give an overview of testing process and major phases of the testing process. Specify what is not covered in the scope of the testing such as, supporting or not third party software.

Glossary
It gives definition of the technical terms used in this document.

References
Any references to other external credentials stated in this document including references to related project documents. They usually refer the System Requirement Specification and the System Design Specification documents.

Overview of document
Describe the contents and organization of the document.

Test plan
A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, and the person who will do each task, and any risks that require contingency planning.

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

Recording of tests
Specify the format to be used to record test results. It should very specifically name the item to be tested, the person

who did the testing, reference of the test process/data and the results expected by the test, the date tested. If a test fails, the person with the responsibility to correct and retest is also documented. The filled out format would be kept with the specific testing schedule. A database could be used to keep track of testing.

Reporting test results
The summary of what has been tested successfully and the errors that still exist which are to be rectified is specified.

Verification testing

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

Integration testing

- Integration test is done on modules or sub-systems.

Validation testing

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

Acceptance and beta testing

- List test plans for acceptance testing or beta testing. During this test, real data is used for testing by the development team (acceptance testing/alpha testing) or the customer (beta testing). It describes how the results of such testing will be reported back and handled by the developers.

3.5.2 User Manual

This document is complete at the end of the software development process.

Different types of user documentation
Users of the system are not of the same category and their requirements vary widely. In order to cater to the need of different class of user, different types of user documentation are required. The following are the various categories of manuals:

- Introductory manual: How to get started with the system?
- Functional description: It describes functionality of the system.
- Reference manual: It gives details about the system facility.
- System administrator guide: How to operate and maintain the system?
- Installation document: How to install the system?

The following is the typical content of User Manual:
Introduction

- Purpose
  - The purpose of this document and its intended audience is stated. If there is more than one intended audience, provide information in this section and direct the reader to the correct section(s) for his/her interest.
- Scope of project
  - The scope includes overview of the product. It explainswho could use the productand overview the services that it provides. Describe limitation of the system. Describe any restrictions on using or copying the software and any warranties or contractual obligations or disclaimers.
- Glossary
  - It defines the technical terms used in this document. Do not assume that the reader is expert.

- References
  - References to other documents cited anywhere in this document including references to related project documents. This is usually the only bibliography in the document.
- Overview of document
  - The contents and organization of the rest of this document are described.

Instructional manual

This section should be divided in the manner that will make it user friendly.

- System usage
  - Provide examples of normal usage. Images of the screen dumps are very useful to provide a look and feel of the product. Provide any necessary background information. On-line help system is very common for systems today. Information on how to use the on-line help system, how to access it may be provided here.

User reference manual

- List of services
  - Provides an alphabetical listing of services provided by the system with references to page numbers in this document where the concerned service is described.
- Error messages and recovery
  - Provides an alphabetical list of all error messages generated by the system and how the user can recover from each of these errors.

Installation information

Installation information is provided including the operating environment.

Maintenance manual

The supplier/developer of the software is sometimes different from the software maintenance agency. In such cases, the criticality of maintenance manual assumes a bigger role. Maintenance manuals provide precise information to keep your product operating at peak performance. Similar to installation manuals, these documents may range from a single sheet to several hundred of pages.

3.6 Different Standards for Documentation

This software documentation standard is used in the organization for uniform practices for documentation preparation, interpretation, change, and revision, to ensure the inclusion of essential requirements of different standards. Sometimes, documentation as per various standards is stated in the contractual agreement between the software vendor and the customer.

This standard will also aid in the use and analysis of the system/sub-system and its software documentation during the system/software life cycle of a software project.Documentation comes in many forms, e.g., specifications, reports, files, descriptions, plans, source code listings, change requests, etc. and can be in electronic or paper form. The Documentation Standard defines various aspects of documentation such as style, format, and the document revision/change process of these documents.

The International Standards, ISO/IEC 12207– Software life cycle process, describes documentation as one of the supporting parallel process of software development process. It may be noted that this standard is not documentation standard but describes the process of documentation during the software development process.

The following are other documentation standards:
ISO/IEC 18019: Guidelines for the design and preparation of user documentation for application software
This standard describes how to establish what information users need, how to determine the way in which that information should be presented to the users, and then how to prepare the information and make it available. It covers both on-line and printed documentation. It describes standard format and style to be adopted for documentation. It gives principles and recommended practices for documentation.

ISO/IEC 15910: Software user documentation process
This standard specifies the minimum process for creating user documentation for software that has a user interface, including printed documentation (e.g., user manuals), on-line documentation, help text and on-line documentation systems.

IEEE 1063: Software user documentation
It provides minimum requirement for structure, information content and format for user documentation. It does not describe the process to be adopted for documentation. It is applicable for both printed and on-line documentation.

Components of software user documentation are:
- Identification data (e.g., Title Page)
- Table of contents
- List of illustrations
- Introduction
- Information for use of the documentation such as description of software etc.
- Concept of operations
- Procedures
- Information on software commands
- Error messages and problem resolution
- Glossary (to make the reader acquainted with unfamiliar terms)
- Related information sources
- Navigational features
- Index
- Search capability (for electronic document).

Documentation involves recording of information generated during the process of software development life cycle. Documentation process involves planning, designing, developing, distributing and maintaining documents.
During planning phase, documents to be produced during the process of software development are identified. For each document, the following items are addressed:
- Name of the document
- Purpose
- Target audience
- Process to develop, review, produce, design and maintain.

The following form part of activities related to documentation of development phase:
- All documents to be designed in accordance with applicable documentation standards for proper formats, content description, page number, figure/table.
- Source and accuracy of input data for document should be confirmed.
- Use of tools for automated document generation.
- The document prepared should be of proper format. Technical content and style should be in accordance to documentation standards.

Production of the document should be carried out as per the drawn plan. Production may be in either printed form or electronic form. Master copy of the document is to be retained for future reference.

Maintenance
As the software changes, the relevant documents are required to be modified. Documents must reflect all such changes accordingly.

3.7 Documentation and Quality of Software

Inaccurate, incomplete, out of date, or missing documentation is a major contributor to poor software quality. That is why documentation and document control has been given due importance in ISO 9000 standards, SEI CMM software Maturity model. In SEI CMM Process Model and assessment procedure, the goal is to improve the documentation process that has been designed. A maturity level and documentation process profile is generated from the responses to an assessment instrument.

One basic goal of software engineering is to produce the best possible working software along with the best possible supporting documentation. Empirical data show that software documentation products and processes are key components of software quality. Studies show that poor quality, out of date or missing documentation are a major cause of errors in software development and maintenance. Although everyone agrees that documentation is important, not everyone fully realizes that documentation is a critical contributor to software quality.

Documentation developed during higher maturity levels produces higher quality software.

3.8 Good Practices for Documentation

A good practice for documentation includes:

Documentation is the design document
The time to document is before actually implementing any design. A lot of effort can be saved in such cases.

Good documentation projects the quality of software
Many people take poor, scanty or illiterate documentation for a program as a sign that the programmer is sloppy or careless of potential users’ needs. Good documentation, on the other hand, conveys a message of intelligence and professionalism. If your program has to compete with other programs, better make sure that your documentation is at least as good as your competitors.

Summary

- Documentation is a process to help users of software and other people to use and interact with system.
- Document communicates the details about the system targeted at different audience. It explains the system.
- System requirement specification is a set of complete and precisely stated properties along with the constraints of the system that the software must satisfy.
- Document communicates the details about the system targeted at different audience. It explains the system.
- The very first step of any documentation process is to acquire the required source material for preparation of document. The material is collected including specifications, formats, screen layouts and report layouts.
- The SRS should specify only the external system behaviour and not the internal details. It also specifies any constraints imposed on implementation.
- Documentation involves recording of information generated during the process of software development life cycle. Documentation process involves planning, designing, developing, distributing and maintaining documents.
- In SEI CMM Process Model and assessment procedure, the goal is to improve the documentation process that has been designed.
- Inheritance diagram is a design diagram work product that primarily documents the inheritance relationships between classes and interfaces in object-oriented modeling.
- Pseudocode is a kind of prepared English for describing algorithms in an easily readable and modular form.
- Empirical data show that software documentation products and processes are key components of software quality.
- A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities.

References

- Prof. Desai, H.K., Gayatri, S.K., 2008.*Mcs-014 Systems Analysis and Design*, Dotcom Publications.
- Ravichandran, D., 2001. *Introduction to Computers and Communication*, Tata McGraw-Hill Education.
- *File Documentation* [Pdf] Available at: <http://foi.nuim.ie/documents/filing_guidelines.pdf>[Accessed 28 June 2013].
- *GNOME Documentation System*[Pdf] Available at: <https://developer.gnome.org/gdp-handbook/stable/gnomedocsystem.html.en> [Accessed 28 June 2013].
- *Introduction to Document Management Software* [Video online] Available at: <http://www.youtube.com/watch?v=Cf-tKLgYwMo > [Accessed 28 June 2013].
- *Superior Office Systems Document Management Introduction.mp4* [Video online] Available at: <http://www.youtube.com/watch?v=inFBEnt-ppk> [Accessed 28 June 2013].

Recommended Reading

- William, B.G.,1993.*Introduction to Electronic Document Management Systems*, Academic Press Incorporated.
- Harry, K.,1976, *Systems Design End Documentation: An Introduction to the HIPO Method*, Van Nostrand Reinhold Company.
- O'Brien, J.A., 2004.*Introduction To Information Systems*, Tata McGraw-Hill Education.

Self Assessment

1. SEI CMM is a ________ model.
    a. Capability Maturity
    b. Test plan
    c. Document
    d. maturity level

2. _________ is used in the organization for uniform practices for documentation.
    a. Software documentation standard.
    b. Hardware documentation standard.
    c. File documentation
    d. Test plan

3. The system design specification is developed in ____________ process.
    a. single stage
    b. three stage
    c. six stage
    d. two stage

4. Pseudo code is used to describe _______________.
    a. test plan
    b. document
    c. algorithms
    d. models

5. Which of the following statements is false?
    a. Document communicates the details about the system targeted at different audience. It explains the system.
    b. In SEI CMM Process Model and assessment procedure, the goal is to improve the documentation process that has been designed.
    c. A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities
    d. International Standard ISO/IEC 12207 is documentation standard.

6. _________________ provides information as to how to operate and maintain the system.
    a. System administration guide.
    b. Documentation guide
    c. Process guide
    d. Software guide

7. ISO standard ISO/IEC 12207:1995 describes documentation as a ____________ activity.
    a. management
    b. test
    c. support
    d. documentation

8. _________ show that software documentation products and processes are key components of software

quality.

a. Software
b. Process
c. Empirical data
d. Information

9. A ___________ is a tree of sub-routines in a program.
   a. structure chart
   b. data
   c. documents
   d. software data

10. _____________is a process to help users of software and other people to use and interact with system.
    a. Data
    b. Management
    c. Documentation
    d. Teat plan

Chapter IV

System Designand Modelling

Aim

The aim of this chapter is to:

- explicate the concept of system design
- explain the importance of system modelling
- elucidate the need for system design

Objectives

The objectives of this chapter are to:

- explicate the evolution of logical design
- analyse the process of system modelling
- enlist the tools specific for system design

Learning outcome

At the end of this chapter, you will be able to:

- define logical database design
- understand the importance of form design
- identify the various process modelling

4.1 Introduction

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

4.2 Logical and Physical Design

Logical Design is the phase of system development life cycle in which system analyst and user develops concrete understanding of the operation of the system. The figure below depicts various steps involved in the logical design. It includes the following steps:

- Designing forms (hard copy and computer displays) and reports, which describe how data will appear to users in system inputs and outputs;
- Designing interfaces and dialogues, which describe the pattern of interaction between users and software;
- Designing logical databases, which describe a standard structure for the database of a system that is easy to implement in a variety of database technologies.

In logical design, all functional features of the system chosen for development in analysis are described independently of any computer platform. Logical design is tightly linked to previous system development phases, especially analysis. The three sub phases mentioned in the figure 4.1 are not necessarily sequential. The project dictionary or CASE repository becomes an active and evolving component of system development management during logical design. The complete logical design must ensure that each logical design element is consistent with others and satisfactory to the end user.

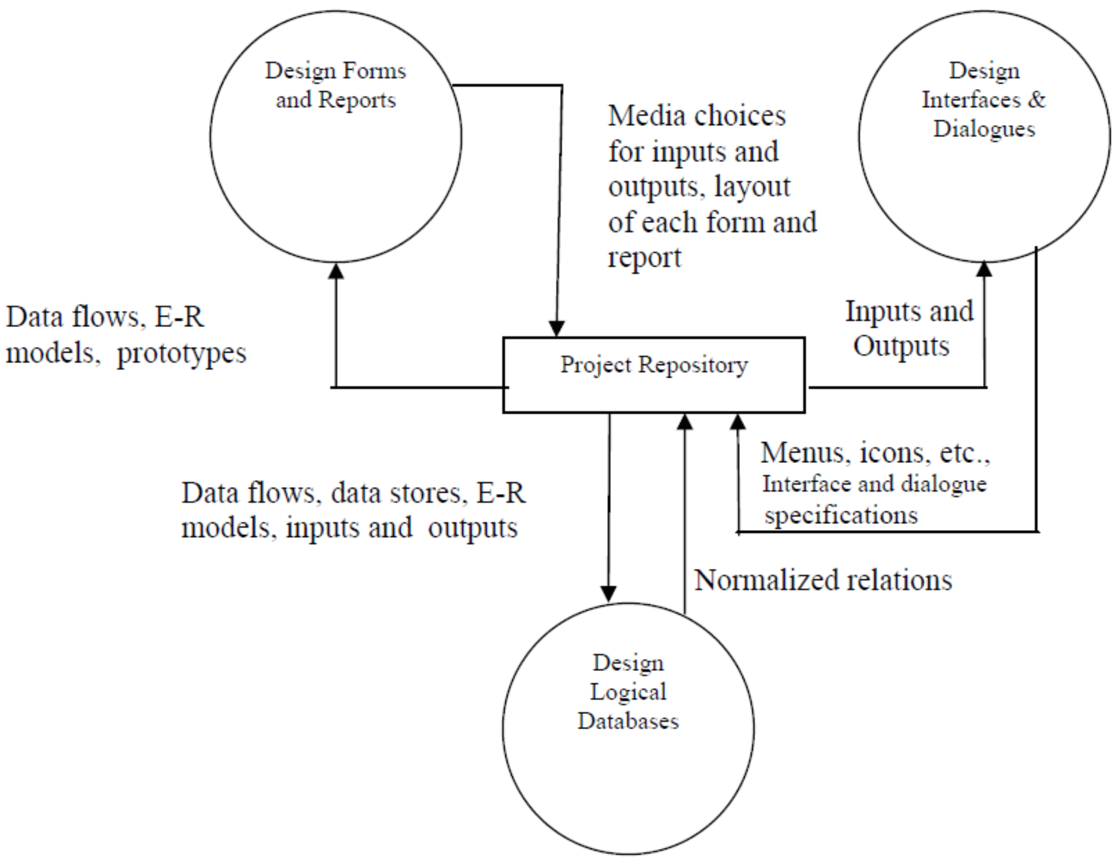

Design Forms and Reports
Design Interfaces & Dialogues
Media choices for inputs and outputs, layout of each form and report
Data flows, E-R models, prototypes
Project Repository
Inputs and Outputs
Menus, icons, etc., Interface and dialogue specifications
Data flows, data stores, E-R models, inputs and outputs
Normalized relations
Design Logical Databases

Fig.4.1 Steps in logical design

Form Design

System inputs are designed through forms. The general principles for input design are:

- Capture only variable data.
- Do not capture data that can be calculated or stored in computer programs.
- Use codes for appropriate attributes.
- Include instructions for completing the form.
- Data to enter should be sequenced.

Common GUI controls for inputs

Text box

It can allow for single or multiple lines of characters to be entered.

Radio button

Radio buttons provide the user with an easy way to quickly identify and select a particular value from a value set. A radio button consists of a small circle and an associated textual description those responses to the value choice. Radio buttons normally appear in groups as one radio buttons per value choice.

Check box

It consists of a square box followed by a textual description of the input field for which the user is to provide the Yes/No value.

List box

A list box is a control that requires the user to select a data item's value from the list of possible choices. It is rectangular and contains one or more rows of possible data values. The values may appear as either a textual description or graphical representation.

Dropdown list
It is like a list box, but is intended to suggest the existence of hidden list of possible values for a data item.

Combination box
It is also known as combo box. It combines the capabilities of a text box and list box. A combo box gives the user, the flexibility of entering a data item's value or selecting its value from a list.

Spin box
A spin box is used to allow the user to make an input selection by using the buttons to navigate through a small set of meaningful choices.

The following steps are to be followed during the process of input design are given below:

- Identify system inputs and review logical requirements.
- Select appropriate GUI (Graphical User Interface) controls.
- Design, validate and test inputs using some combination of:
  - Layout tools (e.g., hand sketches, printer/display layout chart, or CASE)
  - Prototyping tools (e.g., spreadsheet, PC DBMS, 4GL)
- If necessary, design the source document.

Reports design
System outputs are designed through Reports. Outputs can be classified according to two characteristics:

- Their distribution inside or outside the organisation and the people who read and use them; and
- Their implementation method.

Types of outputs

- Internal outputs: Internal outputs are intended for the owners of the system and users within the organisation. There are three sub-classes of internal outputs:
  - Detailed reports which present information with little or no filtering or restrictions;
  - Summary reports which categorise information for managers who do not want to go through details; andException reports—Filter data before it is presented to the manager as information.
- External outputs: These are intended for customers, suppliers, partners and regulatory agencies. They usually conclude or report on business transactions. Examples of external outputs are invoices, account statements, paycheques, course schedules, telephone bills, etc.

Implementation methods for outputs
The commonly used output formats are as follows:

- Tabular output: It presents information as rows and columns of text and numbers.
- Zoned output: It places text and numbers into designated areas or boxes of a form or screen.
- Screen output: It is the online display of information on a visual display device, such as CRT terminal or PC monitor.
- Graphic output: It is the use of a picture to convey information in ways that demonstrate trends and relationships not easily seen in tabular output.

The following are various guidelines for output design:

- Computer outputs should be simple to read and interpret.
- The timing of computer outputs is important.
- The distribution of (or access to) computer outputs must be sufficient to assist all relevant system users.
- The computer outputs must be acceptable to the system users who will receive them.

The steps to be followed during process of output design are given below:

- Identify system outputs and review logical requirements.
- Specify physical output requirements.
- As necessary, design any pre-printed external forms.
- Design, validate and test outputs using some combination of:
  - Layout tools (e.g., hand sketches, printer/display layout chart, or CASE).
  - Prototyping tools (e.g., spreadsheet, PC DBMS, 4GL).
  - Code generating tools (e.g., report writer).

User interface design

User interface design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue.

The following are various guidelines for user interface design:

- The system user should always be aware of what to do next.
- The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
- Messages, instructions or information should be displayed long enough to allow the system user to read them.
- Use display attributes sparingly.
- Default values for fields and answers to be entered by the user should be specified.
- A user should not be allowed to proceed without correcting an error.
- The system user should never get an operating system message or fatal error.
- If the user does something that could be catastrophic, the keyboard should be locked to prevent any further input, and an instruction to call the analyst or technical support should be displayed.

Logical Database Design

Data modelling is used for logical database design. A conceptual model of data used in an application is obtained by using an entity relationship model (E-R model). E-R model assists in designing relational databases. A relational database consists of a collection of relations relevant for a specified application. A relation is a table which depicts an entity set. Each column in the relation corresponds to an attribute of the entity. Each row contains a member of the entity set.

Normalisation is a procedure used to transform a set of relations into another set which has some desirable properties. Normalisation ensures that data in the database are not unnecessarily duplicated. It also ensures that addition and deletion of entity rows (or tuples) or change of individual attribute values do not lead to accidental loss of data or errors in database.

The steps to be followed during physical design are given below:

- Designing physical files and databases - describes how data will be stored and accessed in secondary computer memory and how the quality of data will be ensured.
- Designing system and program structure- describes the various programs and program modules that correspond to data flow diagrams and other documentation developed in earlier phases of lifecycle.
- Designing distributed processing strategies- describes how your system will make data and processing available to users on computer networks within the capabilities of existing computer networks.

The figure below depicts various steps involved in physical design.

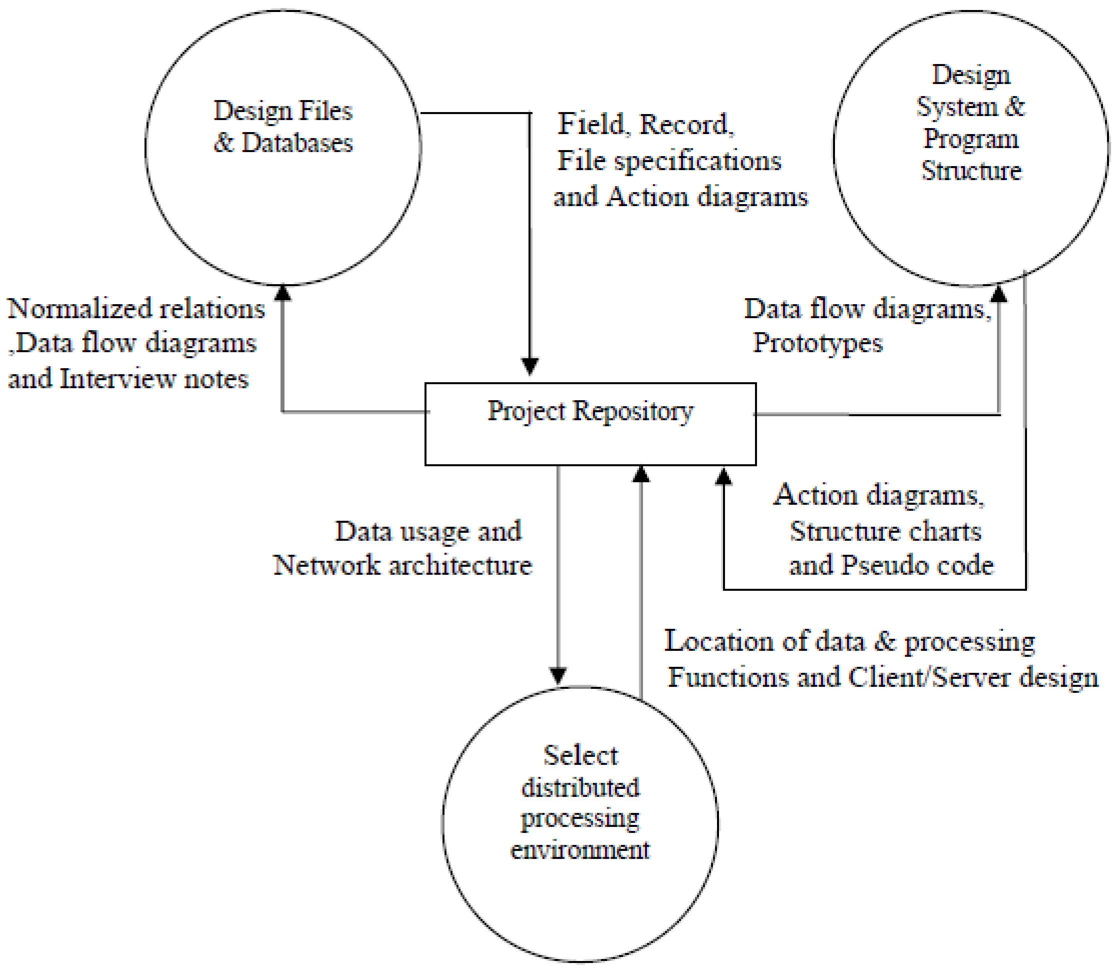

Design Files & Databases
Field, Record, File specifications and Action diagrams
Design System & Program Structure
Normalized relations ,Data flow diagrams and Interview notes
Data flow diagrams, Prototypes
Project Repository
Data usage and Network architecture
Action diagrams, Structure charts and Pseudo code
Location of data & processing Functions and Client/Server design
Select distributed processing environment

Fig. 4.2 Steps in physical design

4.3 Process Modelling

Process modelling involves graphically representing the functions or processes, which capture, manipulate, store and distribute data between a system and its environment and between components within a system. A common form of a process model is data flow diagram. It represents the system overview.

4.3.1 Data Flow Diagrams

A DFD can be categorised in the following forms:

- Context diagram: An overview of an organisational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system. In this diagram, a single process represents the whole system.
- First level DFD: A data flow diagram that represents a system's major processes, data flows, and data stores at a high level of detail.
- Functional decomposition diagram: Functional decomposition is an iterative process of breaking the description of a system down into finer and finer detail which creates a set of charts in which one process on a given chart is explained in greater detail on another chart.

There is no limit on the number of levels of Data Flow Diagrams that can be drawn. It depends on the project at hand.

The following are various components of a Data Flow Diagram:

Process

During a process, the input data is acted upon by various instructions whose result is transformed data. The transformed data may be stored or distributed. When modelling the data processing of a system, it doesn't matter whether the process is performed manually or by a computer. People, procedures or devices can be used as processes that use or produce (transform) data.The notation (given by Yourdon) for process is:

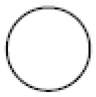


Fig.4.3 Process

Data Flow

Data moves in a specific direction from a point of origin to point of destination in the form of a document, letter, telephone call or virtually any other medium. The data flow is a "packet" of data.The notation (given by Yourdon) for data flow is:

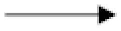


Fig.4.4 Data flow

Source or sink of data

The origin and /or destination of data sometimes referred to as external entities. These external entities may be people, programs, organisation or other entities that interact with the system but are outside its boundaries. The term source and sink are interchangeable with origin and destination.The notation (given by Yourdon) for source or sink is:

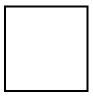


Fig.4.5 Source or sink

Data store

A data store is data at rest, which may take the form of many different physical representations. They are referenced by a process in the system. The data store may reference computerised or non-computerised devices.Notation (given by Yourdon) for data store is:

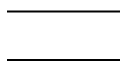


Fig.4.6 Data store

Rules for drawing a data flow diagram:

For process

- No process can have only outputs.
- No process can have only inputs.
- A process has a verb phrase label.

For data store

- Data cannot move directly from one data store to another data store. Data must be moved through a process.
- Data cannot move directly from an outside source to data store. Data must be moved through a process that receives data from the source and places it into the data store.
- Data cannot move directly to an outside sink from a data store. Data must bemoved through a process.

A data store has a noun phrase label.

For source/sink

- Data cannot move directly from a source to a sink. It must be moved by a process
- A source/sink has a noun phrase label

For data flow

- A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read operation before an update.
- A data flow cannot go directly back to the same process it leaves. There must be at least one other process which handles the data flow, produces some other data flow and returns the original data flow to the beginning process.
- A data flow to a data store means update (delete or change).
- A data flow from a data store means retrieve or use.
- A data flow has a noun phrase label.

4.4 Data Modelling

It is a technique for organising and documenting a system's data. Data modelling is sometimes called database modelling because a data model is eventually implemented as database. It is also sometimes called information modelling. The tool for data modelling is entity relationship diagram.

4.4.1 ER Diagram

It depicts data in terms of entities and relationships described by the data. Martin gives the following notations for the components of ERD.

Entities

An entity is something about which the business needs to store data. An entity is a class of persons, places, objects, events or concepts about which we need to capture and store data. An entity instance is a single occurrence of an entity. The notation is given below:

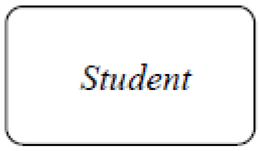


Fig.4.7. Student is the name of entity

Attribute

An attribute is a descriptive property or characteristic of an entity. Synonyms include element, property and field. A compound attribute is one that actually consists of other attributes. It is also known as a composite attribute. An attribute "Address" is the example of compound attribute as shown in the following illustration.

Relationships

A relationship is a natural business association that exists between one or more entities. The relationship may represent an event that links the entities.

The following are important terms related to ER diagrams:

Cardinality
This defines the minimum and maximum number of occurrences of one entity that may be related to a single occurrence of the other entity. Because all relationships are bidirectional, cardinality must be defined in both directions for every relationship. Figure 4.7 depicts various types of cardinality.

Degree
The degree of a relationship is the number of entities that participate in the relationship.

Recursive relationship
A relationship that exists between different instances of the same entity is called recursive relationship. The figure below depicts recursive relationship between the instances of the Course entity.

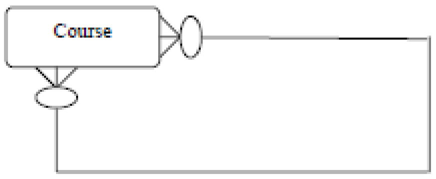


Figure 7.3: Example of Recursive relationship

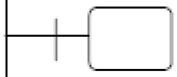
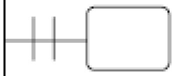
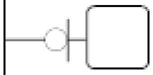
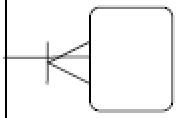
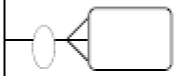
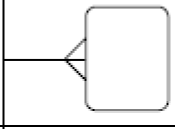
Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one (One and only one)	1	1	 or 
Zero or one	0	1	
One or more	1	Many (>1)	
Zero, one or more	0	Many (>1)	
More than one	>1	>1	

Fig. 4.7 Different types of cardinality

4.5 Process Specification Tools

Processes in Data Flow Diagrams represent required tasks that are performed by the system. These tasks are performed in accordance with business policies and procedures.

Policy
A policy is a set of rules that govern some task or function in the business. Policies consist of rules that can often be translated into computer programs. Systems Analystalong with representatives from the policy making organisation can accurately convey those rules to the computer programmer for programming purposes.

Procedures
Procedures put the policies into action. Policies are implemented by procedures. Procedures represent the executable instructions in a computer program.

There are tools with the help of whom specification for policies can be created. They are Decision Table, Decision Tree and Software Engineering notations

4.5.1 Decision Tables

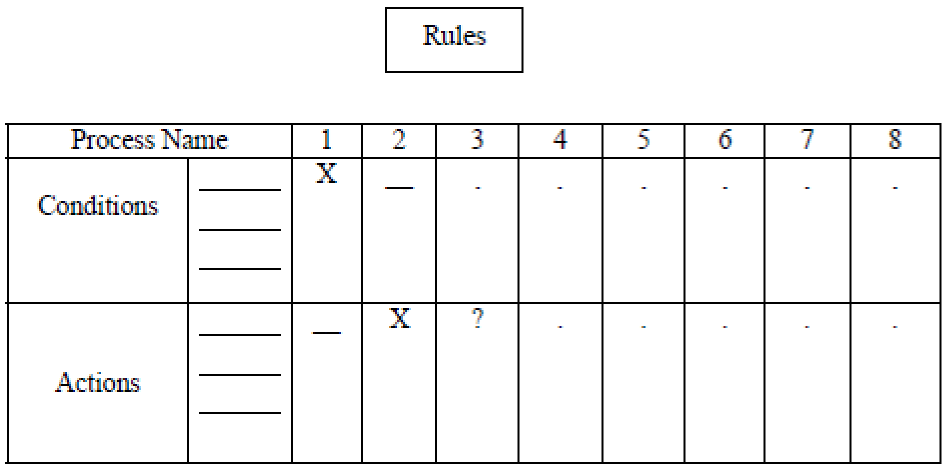
Decision Table is very useful for specifying complex policies and decision-making rules. Figure below depicts a Decision table.

The following are various components of a Decision table:

Condition stubs
This portion of table describes the conditions or factors that will affect the decision or policy making of the organisation.

Action stubs
This portion describes the possible policy actions or decisions in the form of statements.

Rule
Rules describe which actions are to be taken under a specific combination of conditions.Decision tables use a standard format and handle combinations of conditions in a very concise manner. Decision table also provides technique for identifying policy incompleteness and contradictions.


Rules

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

Fig. 4.8 An example decision table

X- Action (condition is true)
_ -Condition is irrelevant for this event.
? - Unknown rule

4.5.2 Decision Trees

Decision tree is a diagram that represents conditions and actions sequentially, and thus shows which conditions to consider first, and so on. It is also a method of showing the relationship each condition and permissible subsequent actions. The diagram resembles branches on a tree.

The root of the tree is the starting point of the decision sequence. The particular branch to be followed depends on the conditions that exist and decision that will be made. Progression from the left to right along a particular branch is the result of making a series of decisions. Following each decision point is the next set of decisions to be considered. The nodes of the tree thus represent conditions and indicate that a determination must be made about which condition exists before the next path can be chosen. The figure below depicts a Decision tree.

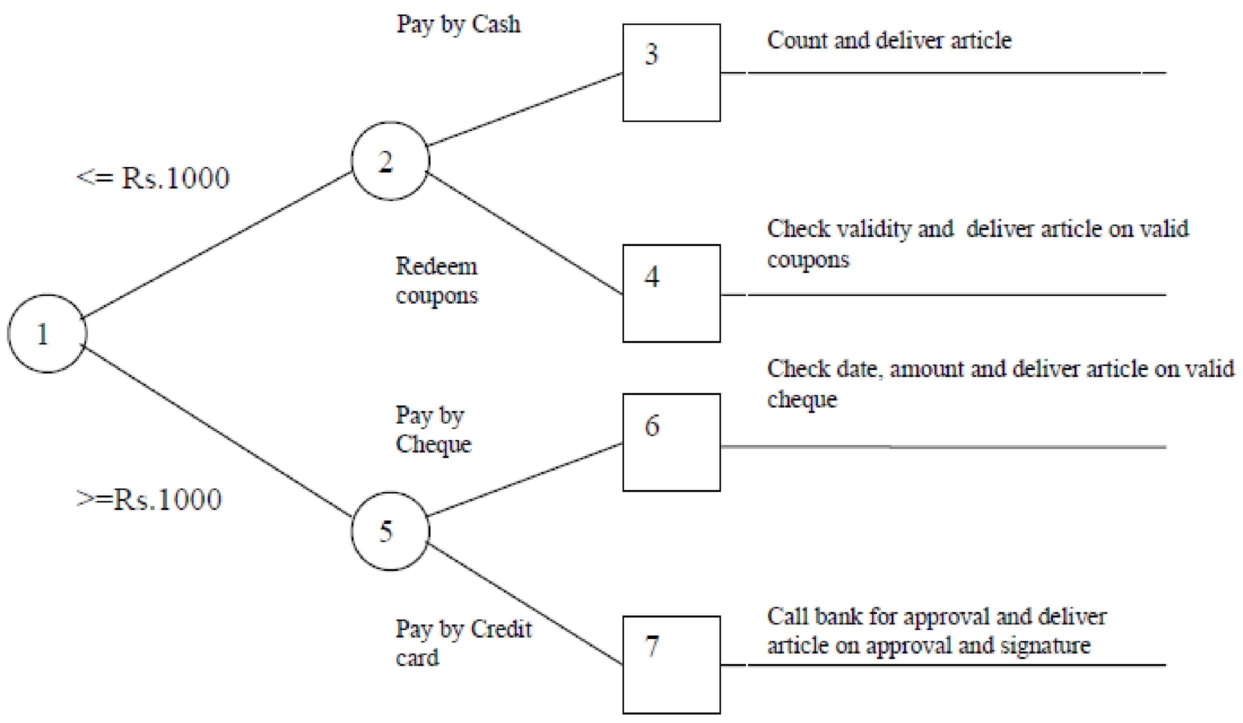

Pay by Cash
3
Count and deliver article
<= Rs.1000
2
Check validity and deliver article on valid coupons
Redeem coupons
4
1
Check date, amount and deliver article on valid cheque
Pay by Cheque
6
>=Rs.1000
5
Pay by Credit card
7
Call bank for approval and deliver article on approval and signature

Fig. 4.9 An Example of Decision tree for a customer bill payment system

4.5.3 Structured English Notation

It is a tool for describing process. There are three valid constructs in this notation. They are:

- A sequence of single declarative statements.
- The selection of one or more declarative statements based on a decision, e.g., if-then-else, switch, case.
- The repetition of one or more declarative statements, e.g., looping constructs such as do-while, for-do

The following are the guidelines usage of Structured English notation:

- Avoid computer programming language verbs such a move, open or close.
- The statement used in the Structured English Notation should always specify the formula to be used.

Structured English notation is based on the principles of structured programming. Process specification logic consists of a combination of sequences of one or more imperative sentences with decision and repetition constructs.
Consider the following:

- An imperative sentence usually consists of an imperative verb followed by the contents of one or more data stores on which the verb operates.
- For example, add PERSONS-SALARY TO TOTAL SALARY
- In imperative sentences, verbs such as "process", "handle" or "operate" should not be used.
- Verbs should define precise activities such as "add" or compute average etc.
- Adjectives that have no precise meaning such as "some" or "few" should also not be used in imperative sentences, because they cannot be used later to develop programs.
- Boolean and arithmetic operations can be used in imperative statements. Table 7.1 lists the arithmetic and Boolean operators.

Arithmetic	Boolean
Multiply (*)	AND
Divide (/)	OR
Add (+)	NOT
Subtract (-)	Greater Than (>)
Exponentiation (**)	Less Than (<)
	Less Than or Equal To (<=)
	Greater Than or Equal to (>=)
	Equals to (=)
	Not equal to (≠)

Table 4.1 Arithmetic and Boolean Operators

Structured English logic:
Structured English uses certain keywords to group imperative sentences and define decision branches and iterations. These keywords are:
(BEGIN, END), (REPEAT, UNTIL), (IF, THEN, ELSE), (DO, WHILE), FOR, CASE, etc.

Grouping imperative sentences

Sequence construct
A sequence of imperative statements can be grouped by enclosing them with BEGIN and END keywords.

Decision (Selection)

- A structure, which allows a choice between two groups of imperative sentences. The key words IF, THEN and ELSE are used in this structure. If a condition is 'true', then group 1 sentences are executed. If it is false, then group 2 sentences are executed.
- A structure which allows a choice between any numbers of groups of imperative sentences. The keywords CASE and OF are used in this structure. The value of a variable is first computed. The group of sentences that are selected for execution depends on that value.

Repetition
This structure shows two ways of specifying iterations in structured English.

- One way is to use the WHILE…DO structure. Here, the condition is tested before a set of sentences is processed.

Alternative to WHILE…DO is FOR structure.

- REPEAT…UNTIL structure. Here, a group of sentences is executed first then the condition is tested. So, in this structure, the group of sentences is executed at least once.

Table 4.2 depicts the criteria to be used for deciding the notation among Structured English, Decision Tables and Decision Trees. Table 4.3 depicts the criteria to be used for deciding the notation among Decision Tables and Decision Trees.

Criteria	Structured English	Decision Tables	Decision Trees
Determining condition & actions	2	3	1
Transforming conditions & actions into sequence	1	2	1
Checking consistency & completeness	2	1	1

Table 4.2 Criteria for deciding the notation to be used

Criteria	Decision Tables	Decision Trees
Portraying complex logic	Best	Worst
Portraying simple problems	Worst	Best
Making decisions	Worst	Best
More compact	Best	Worst
Easier to manipulate	Best	Worst

Table 4.3 Criteria for deciding the notation to be used between Decision tables and Decision trees

4.6 Data Dictionary

A Data Dictionary consists of data about data. The major elements of data dictionary are data flows, data stores and processes. The data dictionary stores details and descriptions of these elements. It does not consist of actual data in the database. But, DBMS cannot access data in database without accessing data dictionary.

If analysts want to know the other names by which a data item is referenced in the system or where it is used in the system, they should be able to find the answers in properly developed data dictionary. Data dictionaries are hidden from users so that data in it is not tampered.

Analysts use data dictionaries for the following reasons:

- To manage the detail in large systems.
- To communicate a common meaning for all system elements.
- To document the features of the system.
- To facilitate analysis of the details in order to evaluate characteristics and determine changes that should be made to the system.
- To locate errors and omissions in the system.

The dictionary contains two types of descriptions for the data flowing through the system: Data elements and Data structures. Data elements are grouped together to make up a data structure.

Data elements are recorded in data dictionary at the fundamental data level. Each item is identified by a data name, description, alias and length and has specific values that are permissible for it in the system.

A data structure is a set of data items that are related to one another and then collectively describe a component in the system. Data is arranged in accordance with one of the relationships namely sequence, selection, iteration and optional relationship.

Summary

- System Design is the specification or construction of a technical, computer based solution for the business requirements identified in system analysis phase.
- Databases are designed with the help of data modelling tools (for example, E-R diagrams) and computer processes are designed with the help of Structured English notation, Decision Trees and Decision Tables.
- A Data Dictionary consists of data about data. The major elements of data dictionary are data flows, data stores and processes.
- Data Modelling is a technique for organising and documenting a system's data. Data modelling is sometimes called database modelling because a data model is eventually implemented as database.
- Logical Design is the phase of system development life cycle in which system analyst and user develops concrete understanding of the operation of the system.
- Radio buttons provide the user with an easy way to quickly identify and select a particular value from a value set.
- The dictionary contains two types of descriptions for the data flowing through the system: Data elements and Data structures. Data elements are grouped together to make up a data structure.
- Decision tree is a diagram that represents conditions and actions sequentially, and thus shows which conditions to consider first, and so on.
- A policy is a set of rules that govern some task or function in the business. Policies consist of rules that can often be translated into computer programs.
- A list box is a control that requires the user to select a data item's value from the list of possible choices. It is rectangular and contains one or more rows of possible data values.
- User interface design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs.

References

- Gupta, R.K., 2004. *Formal Methods and Models For System Design: A System Level Perspective,* Springer.
- Yildirim, H., &Osita, D.I., 2001. *The Mechanical Systems Design Handbook: Modelling, Measurement, and Control*, CRC Press.
- *Process Modelling for Information System Design* [Pdf] Available at: <http://courses.ischool.berkeley.edu/i290-1/f08/lectures/ISSD-20081020.pdf> [Accessed 1 July 2013].
- *System models* [Pdf] Available at: <http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch8.pdf > [Accessed 1 July 2013].
- *SLPSoft Software System Design and Modelling Version 2013.2* [Video online] Available at: <http://www.youtube.com/watch?v=XuJ3GcMvhHQ> [Accessed 1 July 2013].
- *Information System Analysis and Design - Modelling System Requirements*[Video online] Available at: < http://www.youtube.com/watch?v=urI2aH5fqX0>[Accessed 1 July 2013].

Recommended Reading

- Renevan, L., Gilles, S., 2011. *Integrated Circuit and System Design. Power and Timing Modelling, Optimisation, and Simulation: 20th International Workshop*, Springer.
- Jean, M.B. & Jacques, R., 1997. *Models in System Design*, Springer.
- Dennis, M. B., 2011. *The Engineering Design of Systems: Models and Methods*, John Wiley & Sons.

Self Assessment

1. __________provide the user with an easy way to quickly identify and select a particular value from a value set.
    a. Text box
    b. Check box
    c. List box
    d. Radio buttons

2. A ________ consists of data about data. The major elements of data dictionary are data flows, data stores and processes.
    a. Data Modelling
    b. Data Dictionary
    c. Directory
    d. Decision tree

3. ___________ is a diagram that represents conditions and actions sequentially.
    a. Decision tree
    b. Data modelling
    c. E-R Modelling
    d. Directory

4. A ___________ is a control that requires the user to select a data item's value from the list of possible choices.
    a. text box
    b. list box
    c. check box
    d. data box

5. __________ design is concerned with the dialogue between a user and the computer.
    a. Logical
    b. Physical
    c. Data
    d. User interface

6. A _________ is a set of rules that govern some task or function in the business.
    a. procedures
    b. rules
    c. policy
    d. directory

7. A __________ is a set of data items that are related to one another and then collectively describe a component in the system.
    a. data structure
    b. data directory
    c. text box
    d. clear box

8. Which of the following statements is false?

a. A data structure is a set of data items that are related to one another and then collectively describe a component in the system.
b. Logical design is concerned with the dialogue between a user and the computer.
c. Decision tree is a diagram that represents conditions and actions sequentially.
d. A policy is a set of rules that govern some task or function in the business.

9. ____________ is very useful for specifying complex policies and decision-making rules.
a. Decision Table
b. Decision diagram
c. Data structure
d. Data directory

10. The degree of a __________is the number of entities that participate in the relationship.
a. directory
b. information
c. design
d. relationship

Chapter V

Structured System Design

Aim

The aim of this chapter is to:

- explicate the concept of system design
- explain the importance of design considerations
- elucidate the need for system design

Objectives

The objectives of this chapter are to:

- explain the evolution of logical design
- analyse the process of system flexibility
- enlist the tools specific for processing techniques

Learning outcome

At the end of this chapter, you will be able to:

- define logical design methodologies
- understand the importance of design process
- identify the various system interface specifications

5.1 Introduction

The systems objectives outlined during the feasibility study serve as the basis from which the work of system design is initiated. Much of the activities involved at this stage are of technical nature requiring a certain degree of experience in designing systems, sound knowledge of computer related technology and thorough understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system cannot be designed in isolation without the active involvement of the user. The userhas a vital role to play at this stage too.

As we know that data collected during feasibilitystudy will be utilised systematically during the system design. It should, however, be kept in mind that detailed study of the existing system is not necessarily over with the completion of the feasibility study. Depending on the plan of feasibility study, the level of detailed study will vary and the system design stage will also vary in the amount of investigation and at still needs to be done.

This investigation is generally an urgent activity during the system design as the designer needs to study minute's details in all aspects of the system. Sometimes, but rarely, this investigation may form a separate stage between Feasibilitystudy and ComputerSystem Design. Designing a new system is a creative process which calls for logical as well as lateral thinking. The logical approach involves systematicmoves towards the end-productkeeping in mind the capabilities of the personnel and the equipment at each decision making step. Lateral thought implies encompassing of ideas beyond the usual functions and equipment. This is to ensure that no efforts are being made to fit previous solutions into new situations.

5.2 System Design Considerations

The system design process is not a step by step adherence of clear procedures and guidelines, though, certain clear procedures and guidelines have emerged in recent days, but still much of design work depends on knowledge and experience of the designer. When designer starts working on system design, he will face different type of problems. Many of these will be due to constraints imposed by the user or limitations of the hardware and software available in the market. Sometimes, it is difficult to enumerate the complexity of the problems and solutions thereof since the variety of likely problems is so great and no solutions are exactly similar.

5.2.1 Design Objectives

The primary objective of the &sign; of course, is to deliver the requirements as specified in the feasibility report In general, the following design objectives should be kept in mind:

Practicality
The system must be stable and can be operated by people with averageintelligence.

Efficiency
This involves accuracy, timeliness and comprehensiveness of thesystem output.

Cost
It is desirable to aim for a system stem with a minimum cost subject to the condition that it must satisfy all the requirements.

Flexibility
The system should be modifiable depending on the changing needs of the user. Such modifications should not entail extensive reconstructing or recreation of software. It should also be portable to different computer systems.

Security
This is very important aspect of the design and should cover areas ofhardware reliability, fall back procedures, physical security of data and provision for detection of fraud and abuse.

System design involves fist logical design and then physical construction of the system. The logical design describes the structure and characteristics of features, like the outputs, inputs, files, databases and procedures. The physical construction, which follows the logical design, produces actual program software, files and a working system.

5.2.2 Constraints

The designer normally will work under following constraints:

Hardware
The existing hardware will obviously affect the system design.

Software
The available software (operating system, utilities, language etc.) in the market will constrain the design.

Budget
The budget allocated for the project will affect the scope and depth of design.

Timescale
The new system may be required by a particular time (e.g. the start of a financial year). This may put a constraint on the designer to find the best design.

Interface with other systems
The new system may require some data from another computerised system or may provide data to another system in which case the files must becompatible in format and the system must operate with a certain processingcycle.

5.2.3 Processing Techniques

The processing options available to the designers are:

- Batch processing
- Real-time processing
- On-line processing

It is quite interesting to note, however, that a combination of these is often found to be ideal in additional data processing applications.This increases through-put of the system as also brings down the response time of on-line activities. In most of the business applications, 24 hour data is acceptable enough and hence it is possible to update voluminous data after office-hours in batch mode.

5.2.4 Operation

Typically, the flow of data through a system has been shown in the figure below. Throughout thedesign process as described in the next section, the system designer must consider and specify the requirements of each of these operational areas.

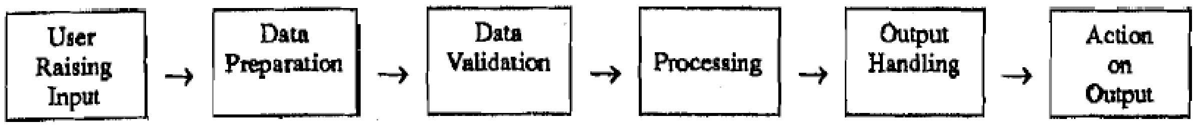


Fig.5.1 Data flow

5.3 Design Methodologies

The scope of the systems &sign is guided by the framework for the new system developedduring analysis. More clearly defined logical method for developing system that meets userrequirements has led to new techniques and methodologies that fundamentally attempt to do the following:

- Improve productivity of analysts and programmers
- Improve documentation and subsequent maintenance and enhancements.
- Cut down drastically on cost overruns and delays
- Improve communication among the user, analyst, designer, and programmer.
- Standardise the approach to analysis and design
- Simplify and sign by segmentation.

5.4 Structured Design

Structured design is a data flow based methodology. The approach begins with a systemspecification that identifies inputs and outputs and describes the functional aspects of the system. The specifications then are used as a basis for the graphic representation. The nextstep is the definition of the modules and their relationships to one another in a form called astructure chart, using a data dictionary and other structured tools.

Logical design proceeds from the top down. General features, such as reports and inputs areidentified first. Then each is studied individually and in more detail. Hence, the structured design partitions a program into small, independent modules. They are arranged in ahierarchy that approximates a model of the business area and is organised in a top downmanner. Thus, structured design is an attempt to minimise the complexity and make aproblem manageable by subdividing it into smaller segments which is called Modularisationor decomposition. In this way, structuring minimises intuitive reasoning and promotes maintainable provovable systems.

A design is said to be top-down if it consists of a hierarchy of modules, with each module having a single entry and a single exit subroutine. The primary advantages of this design areas follows:

- Critical interfaces are tested first.
- Early versions of the design, though incomplete, are useful enough to resemble the real system.
- Structuring the design, parse, provides control and improves morale.
- The procedural characteristics define the order that determines processing.

5.4.1 Major System Design Activities

Several development activities are carried out during structured design. They are data base design, implementation planning, system test preparation, system interface specification, and user documentation.

Database design
This activity deals with the design of the physical database. A keyis to determine how the access paths are to be implemented.

Program design
In conjunction with database design is a decision on the programminglanguage to be used and the flowcharting, coding, and debugging procedure prior to conversion. The operating system limits the programming languages that will run on the system.

System and program test preparation
Each aspect of the system has a separate test ‘requirement. System testing is done after all programming and testing is completed.The test cases cover every aspect of the proposed system, actual operations, and userinterface and so on. System and program test requirements become a part of design specifications a pre requisite to implementation.

In contrast to the system testing is acceptance testing, which puts the system through a procedure design to convince the user that the proposed system will meet the stated requirements. Acceptance testing is technically similar to system testing but politically his different. Acceptance testing is conducted in the presence of the user, audit representatives, or the entire staff.

5.4.2 System Interface Specification

This phase specifies for the user how information should enter and leave the system. The designer offers the user various options. By the end of the design, formats have to be agreed upon so that machine- machine and human-machine protocols are well defined prior to implementation. Before the system is ready for implementation, user documentation in the form of an operator's manual must be prepared. The manual provides instructions on how to install and operate the system, how to provide input, how to access, update, or retrieve information, how to the display or print output, in what format, and so on.

5.4.3 Audit Considerations

A well designed system should have controls to ensure proper operation and routine auditing.A proposed system's failure often results from a lack of emphasis on data control. When designing the system, standards of accuracy, consistency, and maintainability must be specified to eliminate errors and control for fraud. A system design introduces new control elements and changes the control procedures. In a manual system, internal control depends on human judgment, personal care, and division of labour. In a computer-based system, the number of persons involved is considerably reduced. A software packages is an effective substitute for human judgment in processing routines and error checks.

5.4.4 Audit Control and Documentation Control

An important function of system controls is to provide an audit trail. An audit trail is a routine designed to allow the analyst, user, or auditor to verify a process or an area in the new system. In a manual system, the audit trail includes journals, ledgers, and other documents that the auditor uses to trace transactions through the system. In a computerised system, word content and format frequently make it difficult to trace a transaction completely. The systems analyst must be familiar with basic auditing or work closely with an auditor to ensure an effective audit trail during the design phase. For auditing a system in a proper way, documentation is required. Documentation is the basis for the review of internal control by internal or independent auditors. It also provides a reference for system m8intenance, Analyst take lot of time in preparing the documentation.Thus, the aim of auditing is to check that controls built into the design of proposed systems ensure its integrity. Audit considerations must be incorporated at an early stage in the systems development so that changes can be made in time.

5.5 Modularisation

In structure design which is explained above a program is segmented into small,independent modules. These are arranged in a hierarchy that approximates a model of the business area and is organised in a top-down manner with the tails shown at the bottom. Thus, in structured design, we by to minimise the complexity of the problem and make it Manageable by sub-dividing it into smaller segments which is called or decomposition. This has been shown in figure below.

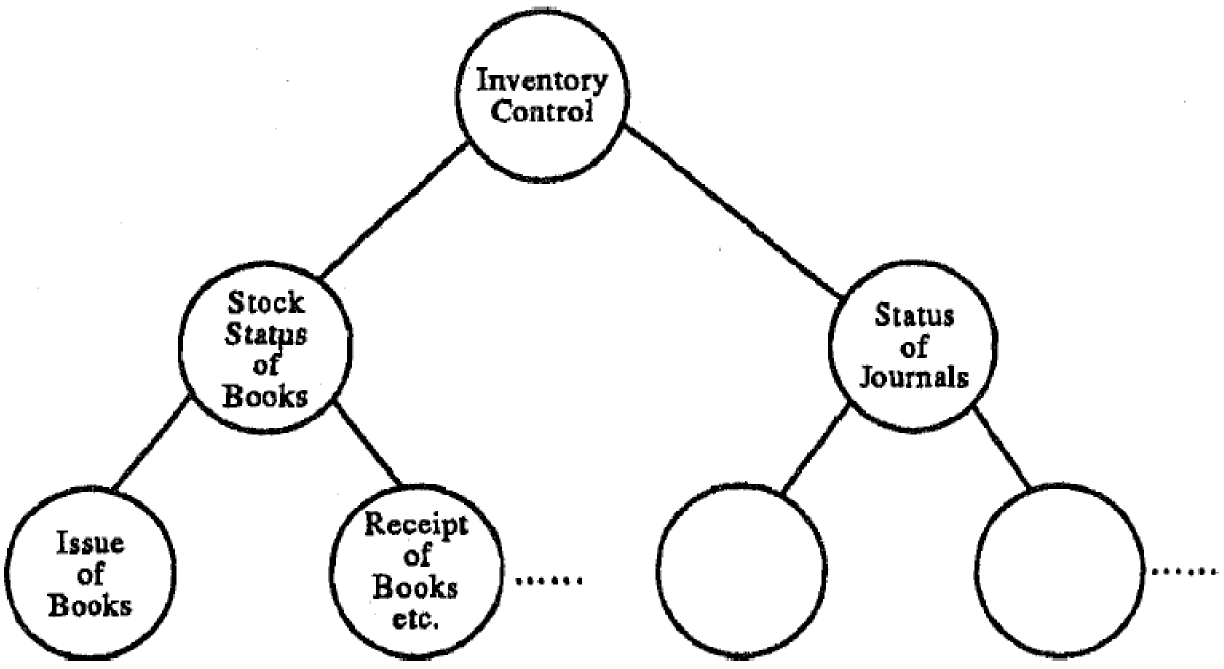


Fig. 5.2 Decomposition a framework

So, the structured design rises from the hierarchical view of the application. The top level asshown in the figure above shows the most important division of work; the lowest level at thebottom shows the details.

5.6 Design Process

The computer system design process is an exercise of specifying "how" the system willwork. It is an iterative process which is based on "what" the system will do as shown in the feasibility report.
Mainly, the following five parts have been included in the system design process:

Output design
The starting point of the design process is the proper knowledge ofsystem requirements which will normally be converted in terms of output.

Input design
Once the output requirements have been finalised, the next step is to findout what data need to be made available to the system to produce the desired outputs.The basic documents in which these data are available need to be identified. Ifnecessary, these documents may have to be revised or new documents may have to bein traduced.

File design
Once the input data is captured in the system, these may have to be preserved either for a short or long period. These data will generally be stored in files in a logical manner. The designer will have to devise the techniques of storing and retrieving data from these files,

Procedure design
This step involves specifications of how processing will beperformed. In this, there are two aspects:

- Computer procedure
- Non-computer procedure

The computer procedure will specify what functions will be carried out on computer what will be different programs and in what sequence the programs will be run. The non-computer procedure will specify the manual procedures for feeding input data, receiving outputs etc.

Control design

The control design indicates necessary procedures which will ensure correctness of processing, accuracy of data, timely output etc. This will ensure that the system is functioning as per plan.

Generally, these steps as mentioned above are inter-dependent and some of them may have tobe used together and traversed many times unti1l a satisfactory design is prepared. It is just like the situation of "Two-step forward-one step backward" kind. In the Figure 5.3 we have tried to present an ideal situation about the progress of a project. But this situation occurs rarely in day-today life. Most of the time progress of a project takes different shape which is shown in Figure 5.4.

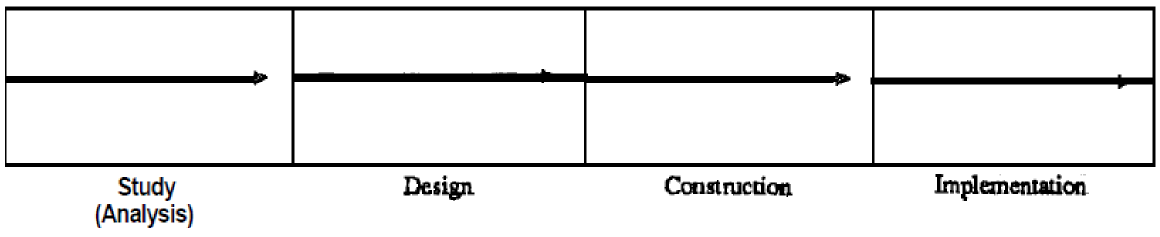


Fig.5.3 "Ideal" project progress

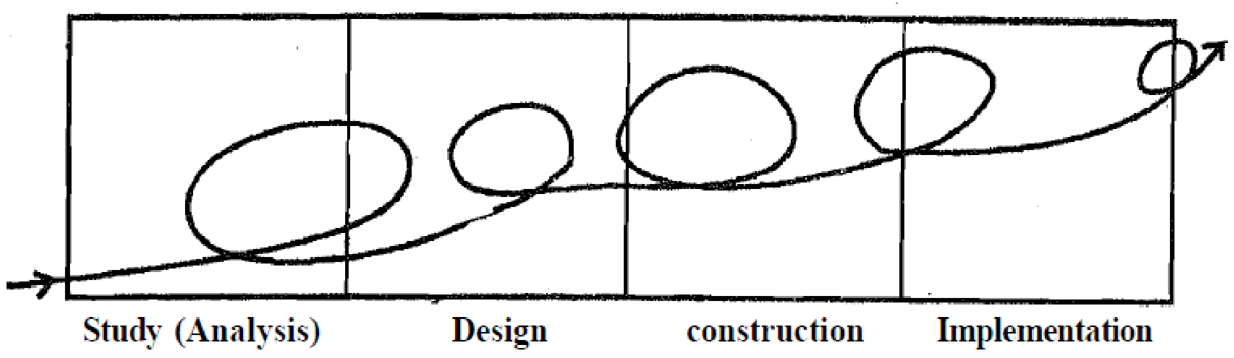


Fig. 5.4 "Reality" of project progress

The system design process is, therefore, an iterative process where decisions made orchanged at one step will have a "ripple effect" on other steps. For example, if an output report is modified, it may necessitate changes in input design, file andsign and control design.

5.7 System Specification

The result of the system design process is a document known as "system specifications".The complete details of &sign about the proposed system have been included in it. It serves as a blue print and helps in developing and implementing the new system. This also formsthe primary documentation on which the system persons will fall back uponafter the system is in use.

Later on, this document is normally divided into different parts foreasy reference. Thus, we get a system manual, user manual, operational manual, since thesystem specifications are prepared as a plan, it becomes sometimes necessary to modify itafter taking into consideration the practical difficulties or bottlenecks or errors found duringlater stages of development. System specifications should include all the details necessary toimplement the system and to understand the whole working of the system.

5.8 Prototype Design

Prototype is a working system that is developed to test ideas and assumptions about the new system. Like any computer-based system, it consists of working software that accepts input, performs some operations on it and gives the output. It is the first version of an information system an original model.

The prototype is actually a pilot at test model. It is designed to be easily changed.Information gained through its use is applied to a modified design that may again be used as a prototype to reveal still better design information. The process is repeated as many times as necessary to reveal essential design requirements. In general, prototypes are considered to be most useful under the following conditions:

- No system with the characteristics of the one proposed has yet been constructed by the developers.
- The essential features of the system are only partially known; others are not identifiableeven through careful analysis of requirements.
- Experience in using the system will significantly add to the list of requirements the, system should meet.
- Alternate versions of the system will evolve through experience and additional development and refinement of its features. The system users will participate in the development process.

The underlying principle of prototyping is as under:

- Users can point out features they like or dislike and so indicate short-comings in an existing and working system more easily than they can describe them in a theoretical or proposed system. Experience and use produce more meaningful comment than analysis of charts and narrative proposals.
- Systems prototyping is an interactive process. It may begin with only a few functionsand be expanded to include others that are identified later. It may also startwith what both analyst and user believes is a complete set of functions that may expandor contract through use and experience.

Typically, these are the steps in the prototyping process:

- Identify the user’s known information requirements and features needed in the system.
- Develop a working prototype.
- Use the prototype, noted needed enhancements and changes. These expand the list of known system requirements.
- Revise the prototype based on information gained through user experience.
- Repeat these steps as needed to achieve a satisfactory system.

As these steps suggest, prototyping is not a trial-and-error development process. Before starting the system design work, user and system analyst sit together and discuss to identifythe requirements. These discussions form the basis for the construction of the prototype. System analyst is fully responsible for the development of the working prototype.

Summary

- The systems objectives outlined during the feasibility study serve as the basis from which the work of system design is initiated.
- Designing a new system is a creative process which calls for logical as well as lateral thinking.
- Prototype is a working system that is developed to test ideas and assumptions about the new system.
- The prototype is actually a pilot at test model. It is designed to be easily changed.Information gained through its use is applied to a modified design that may again be used as a prototype to reveal still better design information.
- The control design indicates necessary procedures which will ensure correctness of processing, accuracy of data, timely output etc.
- The control design indicates necessary procedures which will ensure correctness of processing, accuracy of data, timely output etc.
- The computer system design process is an exercise of specifying "how" the system willwork.
- Documentation is the basis for the review of internal control by internal or independent auditors. It also provides a reference for system m8intenance, Analyst take lot of time in preparing the documentation.
- The systems analyst must be familiar with basic auditing or work closely with an auditor to ensure an effective audit trail during the design phase.
- Thus the aim of auditing is to check that controls built into the design of proposed systems ensure its integrity.
- A well designed system should have controls to ensure proper operation and routine auditing.A proposed system's failure often results from a lack of emphasis on data control.
- Structured design is a data flow based methodology. The approach begins with a systemspecification that identifies inputs and outputs and describes the functional aspects of the system.

References

- Dixit, J.B., 2007. *Structured System Analysis and Design*, Firewall Media.
- Downs, E. & Peter, C., 1992. *Structured Systems Analysis and Design Method: Application and Context,* Prentice Hall.
- *Structured System Design* [Pdf] Available at: <http://www.thedirectdata.com/materials/ce/se/3%20Structured%20System%20Design.pdf> [Accessed 2 July 2013].
- *Structured System Analysis and Design Method* [Pdf] Available at: <http://www.imse.hku.hk/imse2013/ie2013-31.pdf > [Accessed 2 July 2013].
- *What is Structured Cabling Standard (TIA-568-C)?* [Video online] Available at: <http://www.youtube.com/watch?v=NRE6O_mvFus> [Accessed 2 July 2013].
- *System Analysis and Design Lecture 1(Burraq Academy).flv* [Video online] Available at: <http://www.youtube.com/watch?v=F7-C7ci6ojo> [Accessed 2 July 2013].

Recommended Reading

- Kaujalgi, V.B., 1994. *Structured Systems Analysis and Design: Data Flow Approach*, Orient Blackswan.
- Kelkar, S.A., 2004. *Structured Systems Analysis and Design*, PHI Learning Pvt. Ltd.
- Edward, Y., Larry, L.C., 1997. *Structured design: fundamentals of a discipline of computer program and systems design*, Prentice Hall.

Self Assessment

1. The __________ design process is an exercise of specifying "how" the system willwork.
    a. computer system
    b. physical system
    c. logical system
    d. information system

2. __________ is a working system that is developed to test ideas and assumptions about the new system.
    a. Datatype
    b. Logicaltype
    c. Prototype
    d. Systemtype

3. The system design process is, therefore, an iterative process where decisions made orchanged at one step will have a __________ on other steps.
    a. triple effect
    b. data effect
    c. process effect
    d. ripple effect

4. ____________ in using the system will significantly add to the list of requirements the, system should meet.
    a. Experience
    b. Plan
    c. Hardware
    d. Ideas

5. The _________ analyst must be familiar with basic auditing or work closely with an auditor to ensure an effective audit trail during the design phase.
    a. data
    b. hardware
    c. logical
    d. systems

6. ____________ is the basis for the review of internal control by internal or independent auditors.
    a. Interpretation
    b. Documentation
    c. Designing
    d. Reviewing

7. Designing a new system is a creative process which calls for logical as well as __________ thinking.
    a. lateral
    b. plan
    c. ideas
    d. experience

8. Which of the following statements is false?

a. The computer system design process is an exercise of specifying "how" the system willwork.
b. Interpretation is the basis for the review of internal control by internal or independent auditors.
c. Prototype is a working system that is developed to test ideas and assumptions about the new system.
d. Experience in using the system will significantly add to the list of requirements the, system should meet.

9. The __________ design indicates necessary procedures which will ensure correctness of processing, accuracy of data, timely output etc.
   a. documentation
   b. prototype
   c. control
   d. hardware

10. _________ considerations must be incorporated at an early stage in the systems development so that changes can be made in time.
   a. Audit
   b. System
   c. Data
   d. Documentation

Chapter VI

System Requirement Specificationand Analysis

Aim

The aim of this chapter is to:

- introduce Data Flow Diagrams
- elucidate Hierarchy plus Input Process Output
- explain Data Dictionaries

Objectives

The objectives of this chapter are to:

- define the four rules of Data Dictionary
- explain various types of data dictionaries
- explain Construction of a VTOC

Learning outcome

At the end of this chapter, you will be able to:

- recognise the difference between requirements and specifications
- understand the rules of DFD
- identify the symbols used in data dictionary

6.1 Introduction

Requirements and specifications are very important components in the development of any embedded system. Requirements analysis is the first step in the system design process, where a user's requirements should be clarified and documented to generate the corresponding specifications. While it is a common tendency for designers to be anxious about starting the design and implementation, discussing requirements with the customer is vital in the construction of safety-critical systems. For activities in this first stage has significant impact on the downstream results in the system life cycle. For example, errors developed during the requirements and specifications stage may lead to errors in the design stage. When this error is discovered, the engineers must revisit the requirements and specifications to fix the problem. This leads not only to more time wasted but also the possibility of other requirements and specifications errors. Many accidents are traced to requirements flaws, incomplete implementation of specifications or wrong assumptions about the requirements. While these problems may be acceptable in non-safety-critical systems, safety-critical systems cannot tolerate errors due to requirements and specifications. Therefore, it is necessary that the requirements are specified correctly to generate clear and accurate specifications.

There is a distinct difference between requirements and specifications. A requirement is a condition needed by a user to solve a problem or achieve an objective. A specification is a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristics of a system, and often, the procedures for determining whether these provisions have been satisfied. For example, a requirement for a car could be that the maximum speed to be at least 120mph. The specification for this requirement would include technical information about specific design aspects.

Another term that is commonly seen in books and papers is requirements specification which is a document that specifies the requirements for a system or component. It includes functional requirements, performance requirements, interface requirements, design requirements, and development standards. So, the requirements specification is simply the requirements written down on paper.

6.2 Data Flow Diagrams (DFD)

Graphical description of a system's data and how the processes transform the data is known as Data Flow Diagram (or DFD).Unlike detail flowcharts, DFDs do not supply detailed descriptions of modules but graphically describe a system's data and how the data interact with the system.

To construct data flow diagrams, we use:

- arrows
- circles
- open-ended boxes
- squares

An arrow identifies data flow-data in motion. It is a pipeline through which information flows. Like the rectangle in flowcharts, circles stand for a process that converts drawinto information.An open-ended box represents a data/storedata at rest, or a temporary repository of data. A square defies a source (originator) or destination of system data.

The seven rules that govern the construction of data flow diagrams (DFD) are:

- Arrows should not cross each other.
- Squares, circles, and files must bear names.
- Decomposed data flows must be balanced (all data flows on the decomposed diagram must reflect flows in the original diagram).
- No two data flows, squares, or circles can have the same name.
- Draw all data flows around the outside of the diagram.
- Choose meaningful names for data flows, processes, and data stores. Use strong verbs followed by nouns.

- Control information such as record counts, passwords, and validation requirements are not pertinent to a data-flow diagram.

If too many events seem to be occurring at a given point, an analyst can decompose a data conversion (circle). The new data conversions form a parent-child relationship with the original data conversion: the child circle in Figure 6.2 belongs to the parent in Figure 6.1.

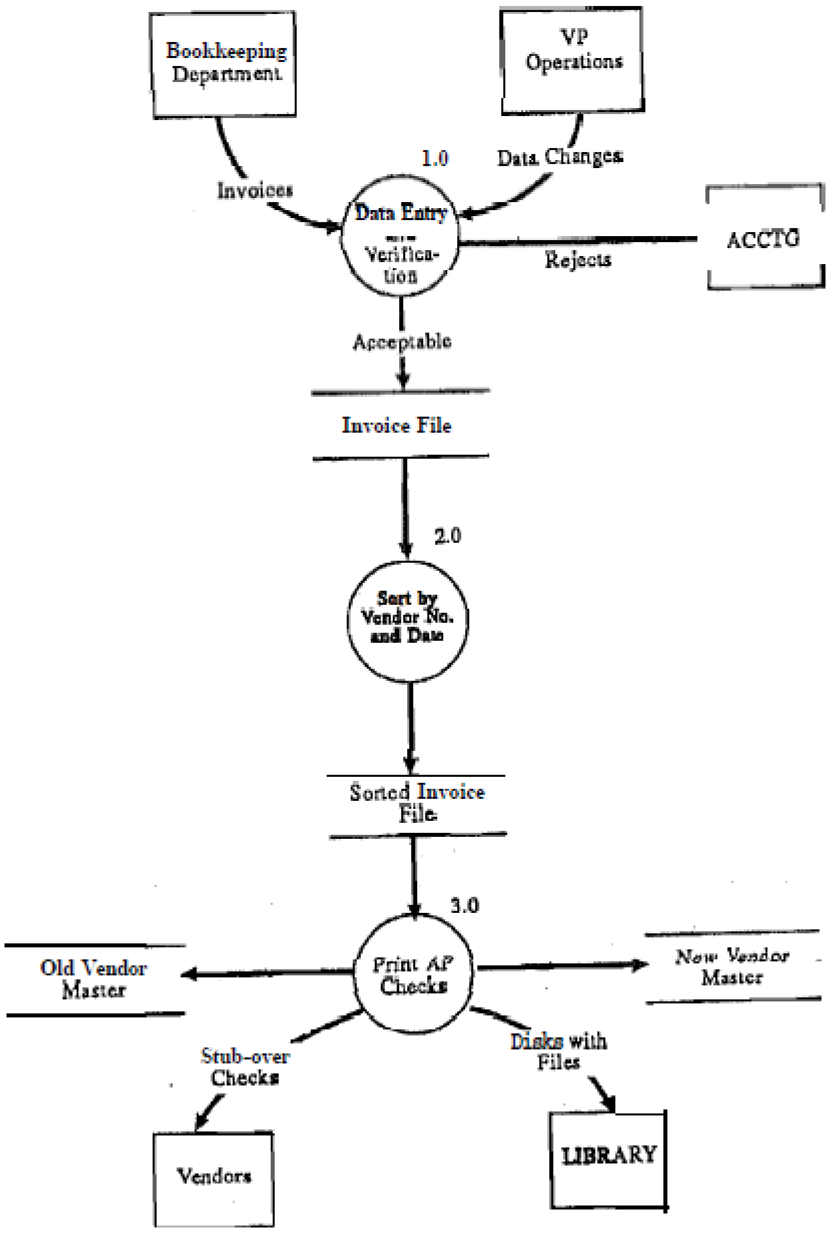


Fig. 6.1 Data conversions: Parent Circle

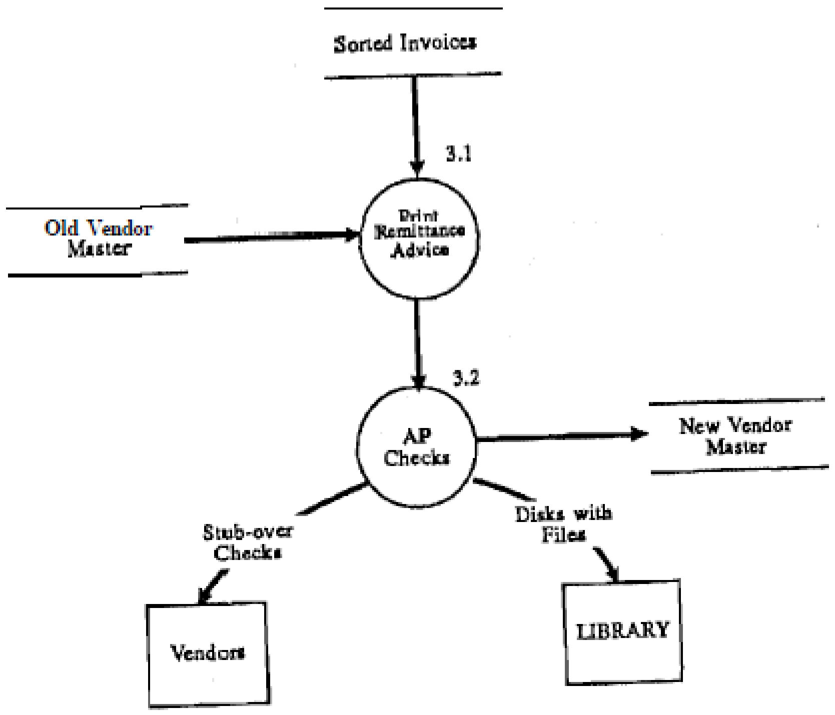


Fig.6.2 Data conversions: Child Circle

Devotes of data-flow, diagrams insist that no other analyst's tool's expresses so fully the flowof data. After all, don't computer people begin with data flow rather than the processing of the data? Another strong advantage is the balancing feature that builds in an error-detection system which the other tools lack. For example, if a parent data-flow diagram shows three inputs and two outputs, the levelled child diagrams taken together must have three inputs and two outputs. If there is an imbalance between the parent and child data-flow diagrams, an error exists in either the parent or the child diagram.

6.2.1 Charting tools used for DFDs

The data flow diagram DFD is the core specification in this method. The figure below shows thevery few charting forms that are necessary.

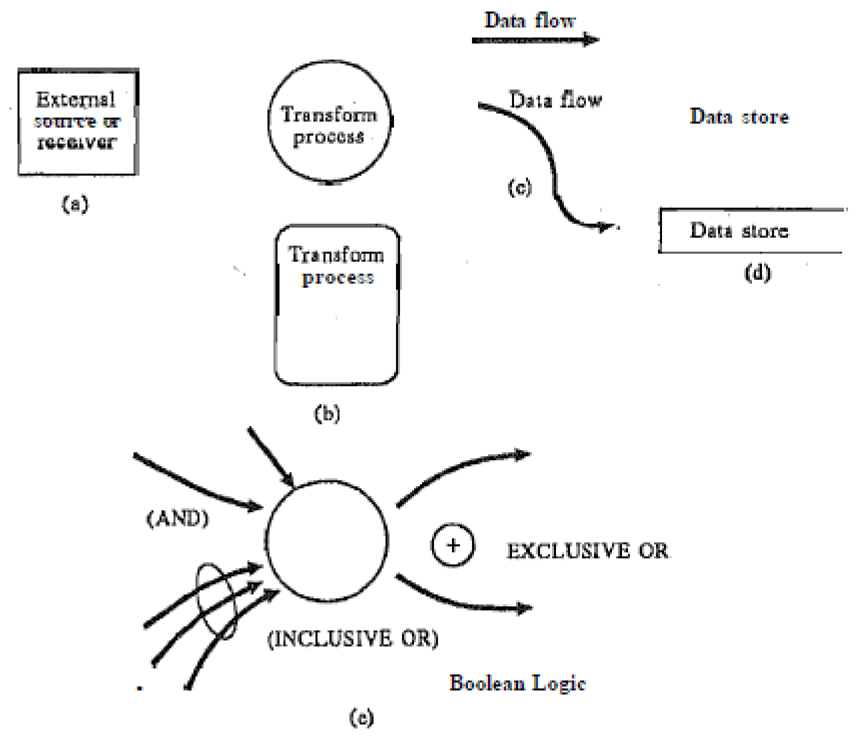


Fig.6.3 Data flow diagram charting forms

The square box shown in Figure 6.3(a) is the external resource or receiver of data and the one shown in Figure 6.3(b) is the transform bubble. Two variations of this have been put forth. The circle or real bubble is the better-known and is used by both Victor Weinberg and Tom De-Marco. The rectangular bubble shown beneath the circle is the form used by Chris Gane andTrish Sarson.

The reason for the rectangular bubble is the perceived need to enter more information than can be contained in the bubble. Tom DeMarco, who is the purist in this group offour writers and educators in the structured method, holds that the data content of the bubblemust be just the bare bones of one verb and a noun, since our objective is not to explain theprocess but to partition into leveled transforms. In fact the rectangular bubble is hard to draw freehand and the template containing this special form is not one you are likely to havearound the shop, it must be specially ordered.

The line arrow is much more important in this method because it carries the data flow, the data into the transform and the data out of the transform. All lines must be identified by their data. Figure 6.3(c) shows the line arrows. The variations on the use of the line arrow are significant, because again the answer needs those different proponents of this method have perceived.

Three different points of view are advanced by the practitioners mentioned above regarding line arrows.

- When multiple lines go into or leave a transform, Weinberg offers the ability to use Boolean logic describing symbols to represent AND, INCLUSIVE OR, and EXCLUSIVE OR. This clearly indicates a felt need for decision logic above the base level. DeMarco advises against using Boolean decision logic. Our examples will not use Boolean logic beyond showing examples in Figure 6.3 (e), since this seems to the author to represent a consensus view all those who use the DFD approach.
- DeMacro shows the line arrow as a curved line giving a different "feeling" than the straight lines andright angles of Weinberg and Gane-Sarson. In the DeMacro approach there is more of a sense of flow thana sense of data in motion. Our examples will use the curved line.

- Regarding the data flow along the line arrow, this would be a serious problem, like the problem of expressing the process within the confines of the bubble. The answer here is to take advantage of the fact that the method allows multiple lines in and out of a bubble and to break up the wordy data flow into several briefly named data flows. It is either that or lengthens the line. This concentration on detail of form; worrying about whether to use a circle or a square or a curved or straight line, may read as petty to the no chartist. To those who mean to use this method to specify systems it is just as serious a matter as the concern of the professional tennis player with the type of racket to be used in a tournament. What we are trying to do with these forms is to invent solutions to problems as we move from the fluid to the concrete and from the tentative to the certain. We need a method for all seasons, but especially to communicate the fluid and the tentative new idea.

The final diagramming element shown in Figure 6.3 (d) is the open rectangle or two parallellines, which indicates the data store (such as a database, file, Kardex or phone book). Ganeand Sarson, unlike the others, show the data store as the two parallel lines joined at one sideto make an open rectangle.

These are all the charting forms we need to use this methodology. Again, as in the flowchartingforms much can be developed out of a few tools. It can be seen that although some practitioners prefer to use variation in their notation, thebroad style is similar.

6.3 Data Dictionaries

A data dictionary defines each term (called a data element) encountered during the analysis and design of a new system. Data elements can describe files, data flows, or processes. For example, suppose you want to print the vendor's name and address at the bottom of a cheque. The data dictionary might define vendor's name and address as follows:

- Vendor name and address =
    - Vendor name +
    - Street +
    - City +
    - State +
    - Pin +
    - Phone +
    - Fax +
    - E: mail

This definition becomes a part of the data dictionary that ultimately will list all key terms used to describe various data flows and files.

6.3.1 Major Symbols

A data dictionary uses the following major symbols:

• =	Equivalent to
• +	And
• [ ]	Either for
• ()	Optional entry

6.3.2 Rules of Data Dictionary

The four rules that govern the construction of data dictionary entries are:

- Words should be defined to stand for what they mean and not thevariable names by which they may be described in the program; use CLIENT-NAME not ABCPQ or CODE06. Capitalisation ofwords helps them to stand out and may be of assistance.
- Each word must be unique; we cannot have two definitions of the same client name.

- Aliases, or synonyms, are allowed when two or more entries show the same meaning; avendor number may also be called a customer number. However, aliases should be usedonly when absolutely necessary.
- Self-defining words should not be decomposed.We can even decompose a dictionary definition. For instance, we might write:

Vendor name = Company name
Individual name
which we might further decompose to:
Company name = (Contact) +
Business name

Individual's name = Last name +
First name+
(Middle initial)

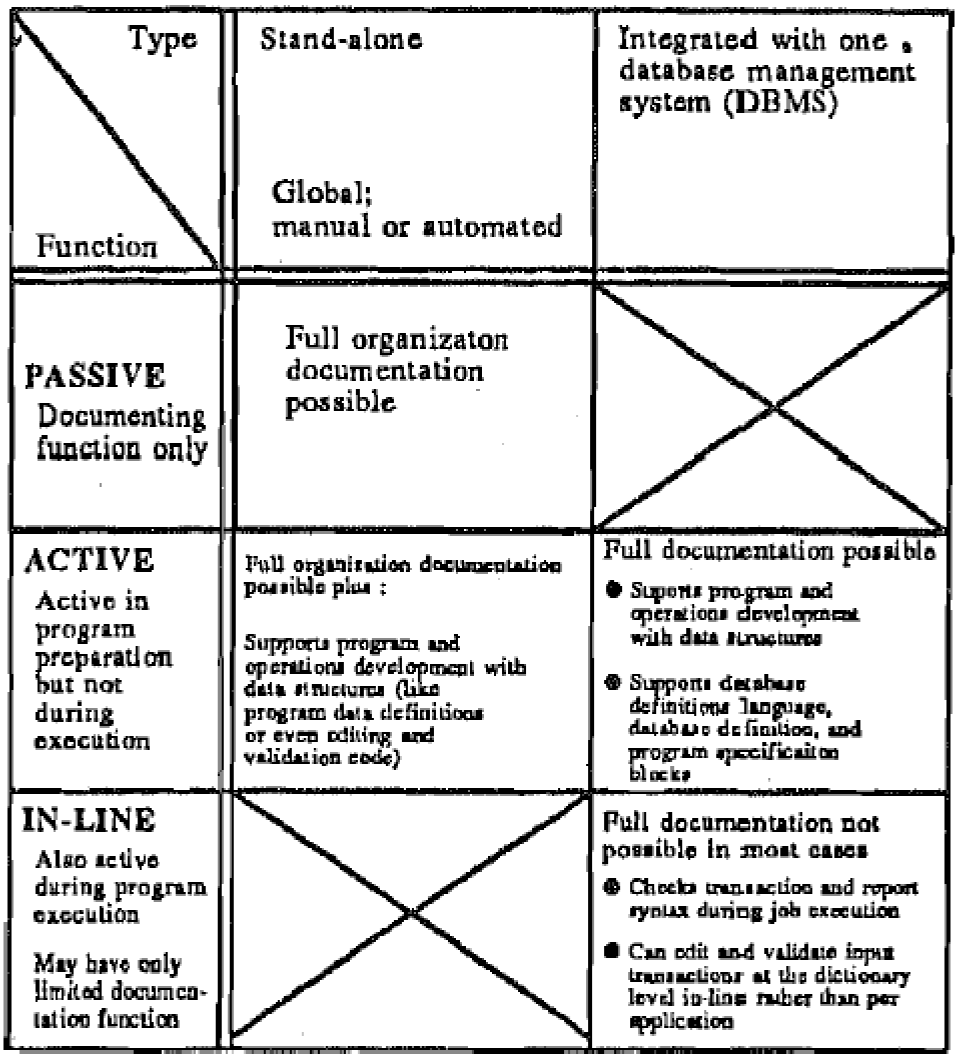
After defining a term, says VENDOR NUMBER, we list any aliases or synonyms, describing the term verbally, signify its length and data type, and list the data stores where the tern is found (figure 6.4). Some terms,may have no aliases, may be found in many files, or may belimited to specific values. Some self-defining or obvious words and terms may not require inclusionin the data dictionary. For example, we all know what a PW code and middle initialare. Data dictionaries seldom include information such as the number of records in, file, thefrequency a process will run, or security factors such as passwords users must enter to gainaccess to sensitive data. Rather, data dictionaries offer definitions of words and termsrelevant to a system, not statistical facts about the system.

Data dictionaries allow analysts to define precisely what they mean by a particular file, dataflow, or process. Some commercial software packages, usually called Data Dictionary Systems(or DDS), help the analysts maintain their dictionaries with the help of the computer. These systems keep track of each term, its definition, which systems or programs use the term,aliases, the number of times a particular term is used and the size of the term can be tied tocommercial data managers.

Data element name	**Vendor-number**
Aliases	None
Description	Unique identifier for vendors in the accounts payable system.
Format	Alphanumeric, six characters.
Data flows	Vendor master Accounts payable open item Accounts payable open adjustments Cheque reconciliation
Reports	Alphabetic vendor list Numeric vendor list A/P transaction register Open item Vendor account inquiry Cash requirements Pre-cheque-writing Cheque register Vendor analysis

Table 6.1 Data Dictionary Systems

Figure 6.5 illustrates the different types of data dictionaries and the functions of each address.



Type / Function	Stand-alone Global; manual or automated	Integrated with one • database management system (DBMS)
PASSIVE Documenting function only	Full organizaton documentation possible	
ACTIVE Active in program preparation but not during execution	Full organisation documentation possible plus : Supports program and operations development with data structures (like program data definitions or even editing and validation code)	Full documentation possible ● Suports program and operations development with data structures ● Supports database definitions language, database definition, and program specificaiton blocks
IN-LINE Also active during program execution May have only limited documentation function		Full documentation not possible in most cases ● Checks transaction and report syntax during job execution ● Can edit and validate input transactions at the dictionary level in-line rather than per application

Figure 6.5 Data-dictionary types and functions

6.3.3 Data Dictionaries

There are two kinds of data dictionaries. They are:

- Integrated:The integrated dictionary is related to one database management system. To the extent, the organisation data is under this DBMS, it is global or organisation wide. However, very few enterprises have all their data eggs in one basket, so the dictionary documentation (metadata) can be considered as local and fragmented
- Stand-alone:The stand-alone dictionary is not tied to any one DBMS, although it may have special advantages for one DBMS, such as the IBM DB-DC Data Dictionary, which has special features related to the IBM IMS DBMS but is still a stand-alone variety of dictionary.

Data dictionary functions

Both these types of dictionaries can be identified by functions as passive, active, or inline. Viewed either way, by type or function, the differences are striking. Passive, active, and in-line dictionaries differ functionally as follows:

Passive data dictionaries
The functionally passive dictionary performs documentation only. This variety of dictionary could be maintained as a manual rather than an automated database. For more than limited documentation use, the automated passive dictionary has clear advantages. From the organisational view the documentation function is the most important dictionary service with the most potential benefits, so the passive dictionary should not be thought of negatively. It has more limited functionality but it may perform its critical function of global documentation.

Active data dictionaries
Besides supporting documentation to one degree or another, the active data dictionary supports program and operations development by exporting database definitions and program data storage definitions for languages such as COBOL and Job Control Language (JCL) forexecution-time performance.

The IBM DB/DC Data Dictionary already mentioned is such a stand-alone, active dam dictionary. A dictionary such as this is not an in-line data dictionary as delivered, which is not to say that it could not be put in-line by a determined effort of major proportions.

In-line data dictionaries
An in-line data dictionary is active during program execution, performing such feats as transaction validation and editing. Such a dictionary would always have some documentation value, but documentation across the organisation about the organisation functions and activities and all the organisation information data stores is not likely. In-line dictionaries are associated with DBMS products such as Cullinet Software Corporation's IDMS-R or Cincom System's TOTAL, to name just two.

6.3.4 The Make-up of Data Dictionaries: Data Dictionary Internals
The minimum data dictionary is shown in figure 6.6.

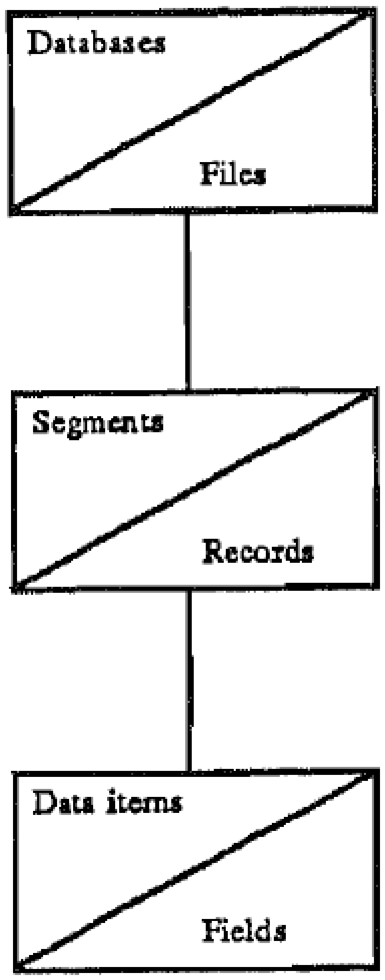


Fig. 6.6 Minimum Data Dictionary

There is a database system consisting of databases or files. These files consist of data groups or segments or records. These data groups consist of data items or fields. There is an implicit relationship here, which needs no additional comment. A certain amount of attribute information is always present. In the case of data items we need to know if it is a or secondary key or an attribute field, if it has aliases, what are the field type and field size, what is the name in various languages, and what is the user description of the item. We need to know whether the data item or data group is in test, system test, or production status. We need to know the number of occurrences of this data item in the dictionary.

Addressing these last points, a data item (for instance) on the IBM data dictionary may look strange to the uninitiated. It will look like this:

T, C, BALANCE-ON-HAND; 0

We recognise balance-on hand as an inventory quantity. The T is the status code, which we will say is T because the data-item balance-on-hand is on the test-data database. The C isthe subject code, which in this case is the primary programming language: COBOL. The 0 is the occurrence number where duplication exists in the common information system. So, in terms of this dictionary, the full description of the data-item consists of the four elements ‘ mentioned above. This convention holds for all subjects defined on the IBM data dictionary. Before discussing the functions of the full-service extended data dictionary we need to review data-dictionary elements. The figure below shows these elements.

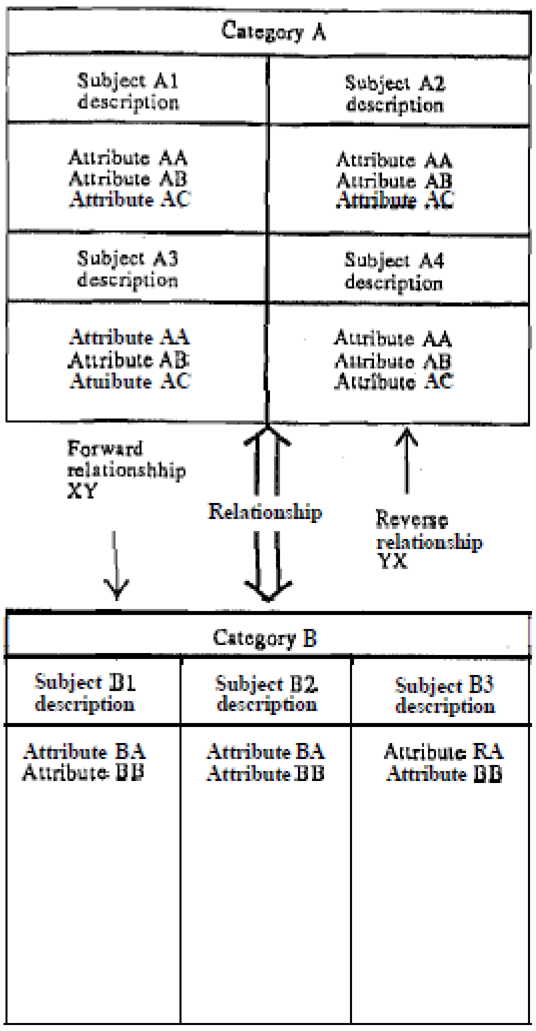


Fig.6.7 Data-dictionary elements

We have already referred to categories, subjects, relationships, attributes, and descriptions on other occasions. These are the elements that make up the data dictionary. In figure 6.7,Category A has a forward and reverse relationship to Category B. We have two-way relationshipssimply because we may want to examine these relationships in both the directions. Data-Item is an example of a category. In a full- service dictionary some categories are predefinedregarding attributes and relationships, but the dictionary has the capacity to handle user defined categories. This, in IBM parlance, is an extended use of the dictionary. This "extensibility" feature is the heart of the full-service dictionary, allowing documentation of thewhole organisation and allowing us to use the dictionary as the software support for strategicand tactical planning.

Category A in figure 6.7 has four subjects. Each subject has the same attribute set as theothers (attributes AA, AB, AC). For instance the category may be Projects. The four subjects are four different projects, described by name and description as unique. Perhaps the attributes are Project Leader, Project Due Date, and Percent Accomplished. All four subjects would have identical attribute names. Perhaps Category B is Information Systems, with subjects and attributes defined in a similar fashion. The forward relationship might be Projects ACCOMPLISH Information Systems; Reverse might be ACCOMPLISHED-BY.

Figure 6.8 shows another example of the elements that make up a data-dictionary database.In this case we have the category Business Function (or department) related to the Processes of the organisation such as Provide-Materials. Remember that the subject name looks like this: P, Provide-Materials, and O. Remember that the P is the status code, which in this case stands for Production. The two adjacent commas means the subject code is not used for this kind of category, and the zero is the occurrence (only this occurrence exists).

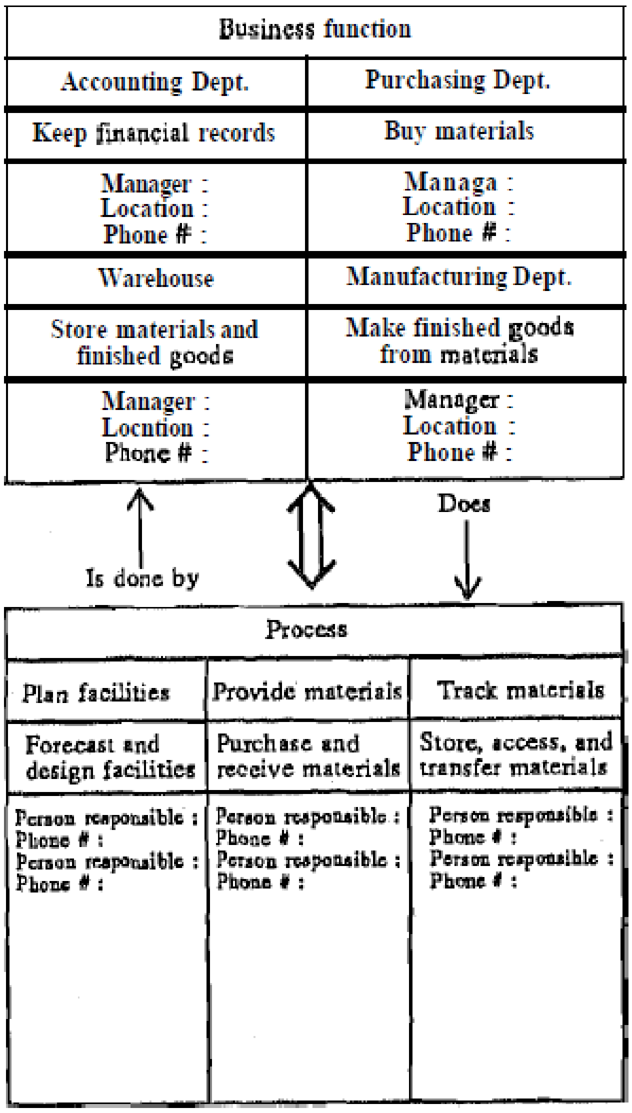


Fig. 6.8 Example of data-dictionary elements

The IBM data dictionary, which is actually a six linked databases, each with many segments, consists of standard categories and the infrastructure needed to "customise" installation categories. The standard categories have the attributes prebuilt and are ready for the user to fill in. These standard categories are:

- Database
- Segment
- Element
- Program communication block
- IMS system definition
- Application system
- Job
- Program
- Module
- Transaction
- PSB

These categories are all related to servicing the data processing function and are not sufficiently broad in scope to support a dictionary for the entire organisation. The strategic plan cannot be documented with just these categories. It is the ability of this data dictionary to allow the creation of other user-defined categories that allows us to consider the dictionary a serious tool for systems analysis and documentation ‘in support of the current and new system applications.

6.4 Hierarchy plus Input Process Output (HIPO)

HIPO stands for Hierarchy plus Input Process Output. It consists of two types of diagrams:

- Visual Table of Contents (VTOC)
- Input Process Output (IPO)

Together these diagrams assist in designing programs and their functions.Following the structured approach that begins with generalities and descends to details, VTOC diagrams break a system or program down into increasingly detailed levels. Therefore, the name of the system appears at the top of the VMC, the names of the major functions within the system lie on the second level and even smaller sub functions lie on the third and succeeding levels.

When used to diagram a program, the VTOC arranges the program modules in order of priority, and it reads from the top down and from left to right. Each module of the program appears as a rectangle which contains a brief description of the module’s purpose (two to four words, beginning with a verb followed by an object i.e. “compute net pay”).

The VTOC for the correct assembly of a bicycle might include five major tasks:

- open the carton,
- remove the parts (that is separate them),
- group similar parts,
- assemble the wheels
- finish assembling the bicycle (figure 6.9)

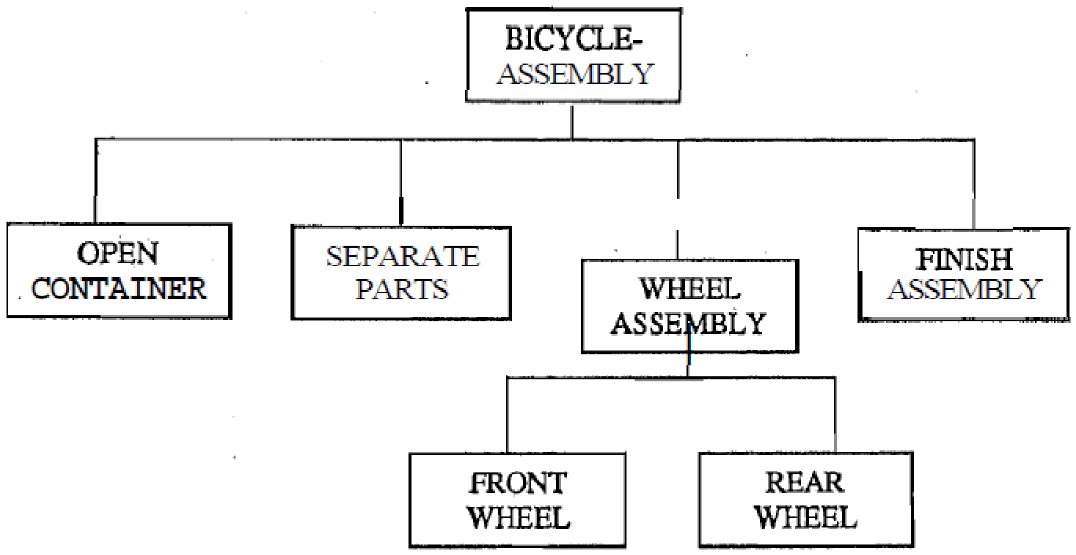


Fig. 6.9 VTOC for the modules to assemble a bicycle

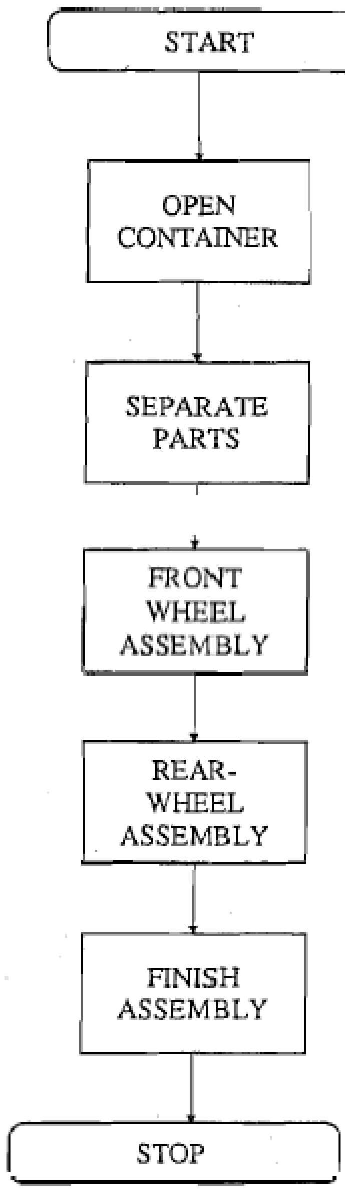
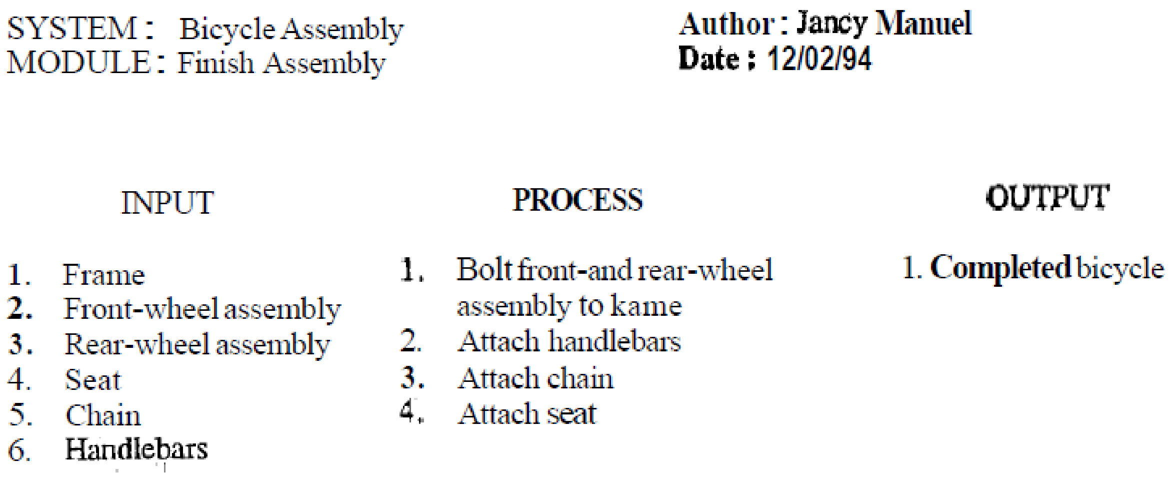


Fig. 6.10 Flowchart of bicycle assembly

Both indicate hierarchy but the VTOC offers a more complete picture of the overall process. When assembling something, no matter how clear the instructions arc, it helps to refer to a picture of the finished product. Within the VTOC, each task must be performed in the order specified, and each task may involve several subtasks. For example, wheel assembly involves the separate subtasks of front land and rear-wheel assembly.

AnIPO chart defines the inputs, processing, and outputs for each module in the program. Figure 6.11 is an IPO for the "finish assembly" module, inputs for which include the frame, front-wheel assembly, rear-wheel assembly, seat, handlebars and chain. The processing requirementsare bolting wheel assemblies to the frame, and attaching handlebars, chain, and seat. The output is the completed bicycle.



SYSTEM : Bicycle Assembly
MODULE : Finish Assembly

Author : Jancy Manuel
Date : 12/02/94

INPUT	PROCESS	OUTPUT
1. Frame	1. Bolt front-and rear-wheel assembly to kame	1. Completed bicycle
2. Front-wheel assembly	2. Attach handlebars	
3. Rear-wheel assembly	3. Attach chain	
4. Seat	4. Attach seat	
5. Chain		
6. Handlebars		

Fig.6.11 Examples of an IPO

6.4.1 Constructing a VTOC

Now, let's learn how to prepare a VTOC for the computerisation of a manual accounts payable system. We first assign all the major outputs (reports) to modules as shown in figure 6.12.

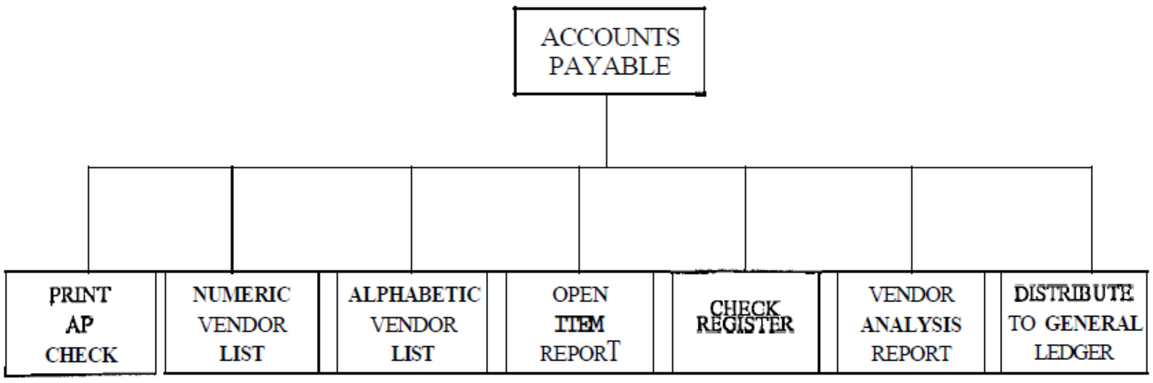


Fig. 6.12 VTOC for an accounts payable system begins with assignment of modules to each output report.

We name modules according to their function so that the reader can tell exactly the purpose of the module (again using the verb-object format). After naming, we choose a number for the system (I), we give each module a sub or level number, beginning with 1 for the most general or highest level function. Therefore, we would identify the overall system as 1.0, the accounts payable check module as 1.1, the numeric vendor list as 1.2 and so on (Figure 6.13). Such a number system clarifies the relationships between modules, and allows anyone reading it to easily locate the detailed IPO charts with corresponding numbers.

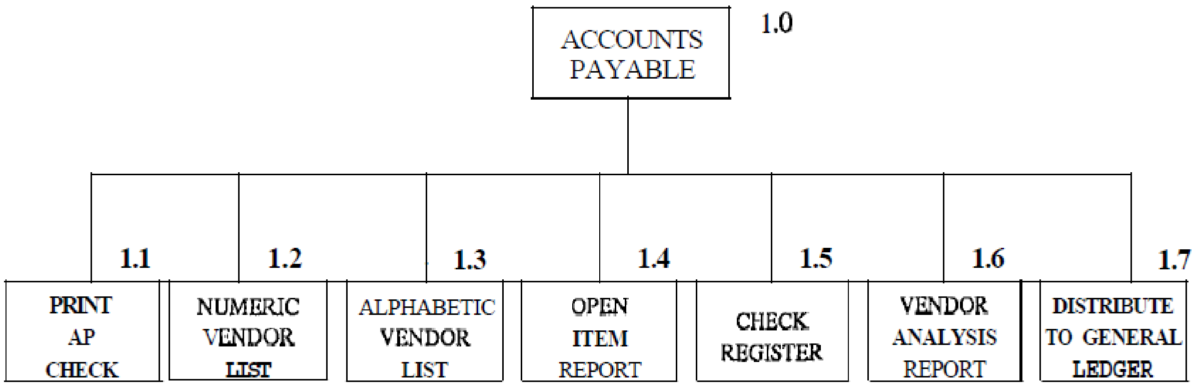
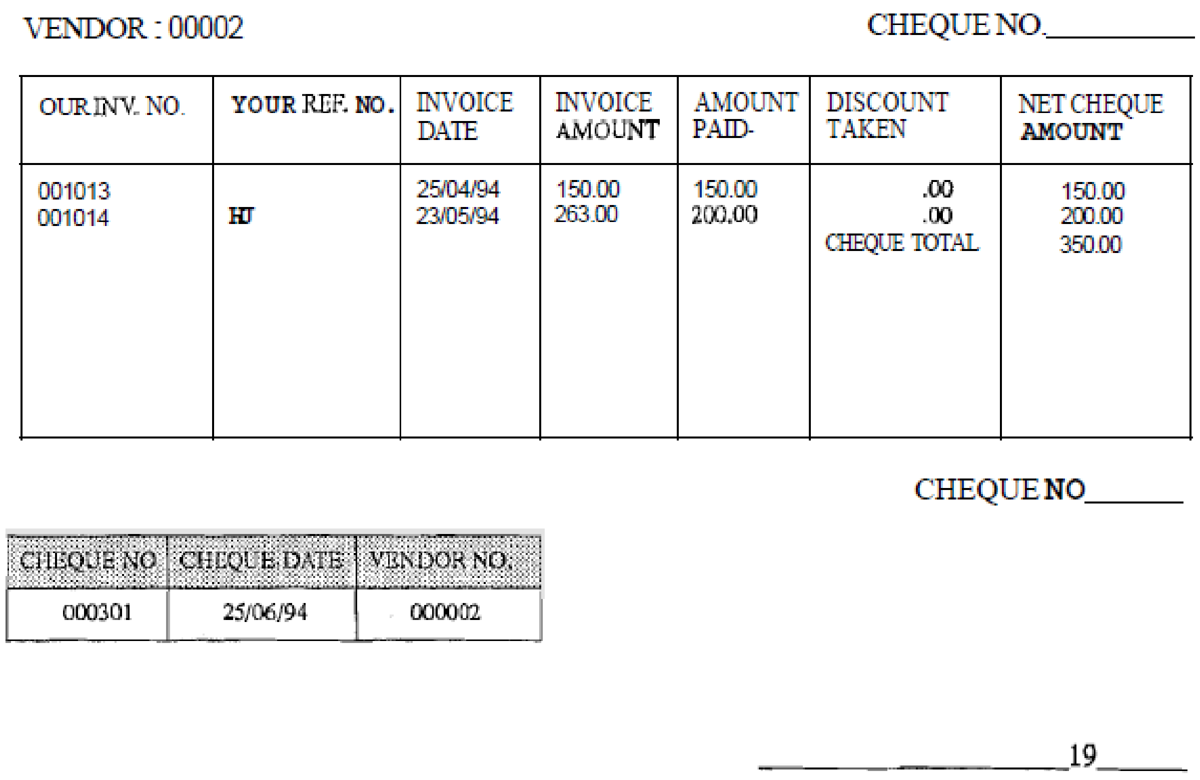


Fig.6.13 The VTOC for the AP check-printing system with level number

After assigning level numbers to each module, we can consider whether further decomposition is necessary. Take the accounts payable check module, for instance. It must be decomposed to a lower level because it represents two tasks (remember, a module must be singlepurpose), one for each part or stub of the check (figure 6.14).



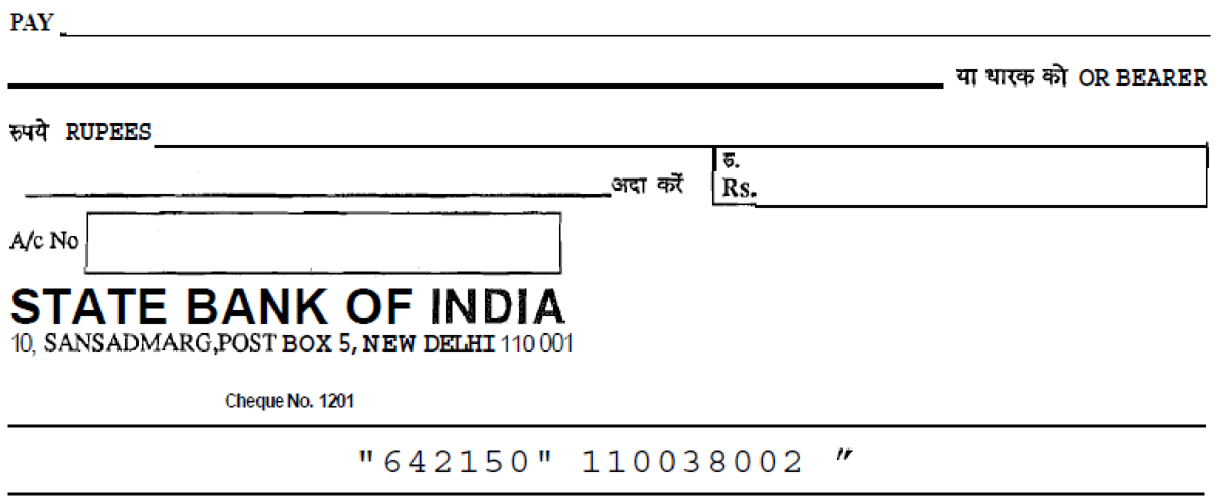
VENDOR : 00002 CHEQUE NO.__________

OUR INV. NO.	YOUR REF. NO.	INVOICE DATE	INVOICE AMOUNT	AMOUNT PAID-	DISCOUNT TAKEN	NET CHEQUE AMOUNT
001013		25/04/94	150.00	150.00	.00	150.00
001014	HJ	23/05/94	263.00	200.00	.00	200.00
					CHEQUE TOTAL	350.00

CHEQUE NO______

CHEQUE NO	CHEQUE DATE	VENDOR NO.
000301	25/06/94	000002

____________________19_____



PAY

या धारक को OR BEARER

रुपये RUPEES

अदा करें

रु.
Rs.

A/c No

STATE BANK OF INDIA
10, SANSADMARG,POST BOX 5, NEW DELHI 110 001

Cheque No. 1201

"642150" 110038002 "

Fig 6.14 (a)Two-part cheque sent to vendors.

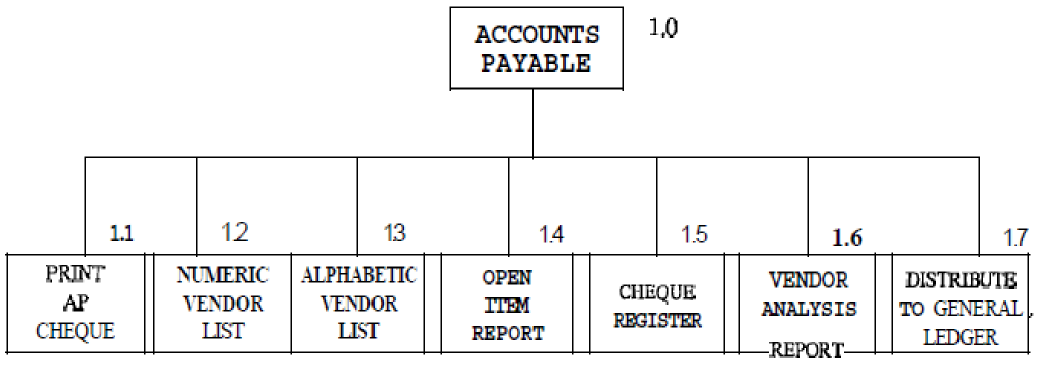


Fig. 6.14 (b) This VTOC reveals the two component modules within the PRINT AP CHECK module.

The upper stub is the remittance advice, which contains the date, number, discount, balance, and total each invoice covered by the cheque. The lower stub is the cheque itself, complete with cheque number, payment amount, and vendor name and number. Applying the top-down concept to the cheque module, we can add another level of modules: one for printing the remittance advice and one for printing the cheque itself (figure 6.15). We assign this third level of modules a third set of numbers (1.1.1 and 1.1.2). Decomposition ends when all modules are single purpose, single entry; and single exit.

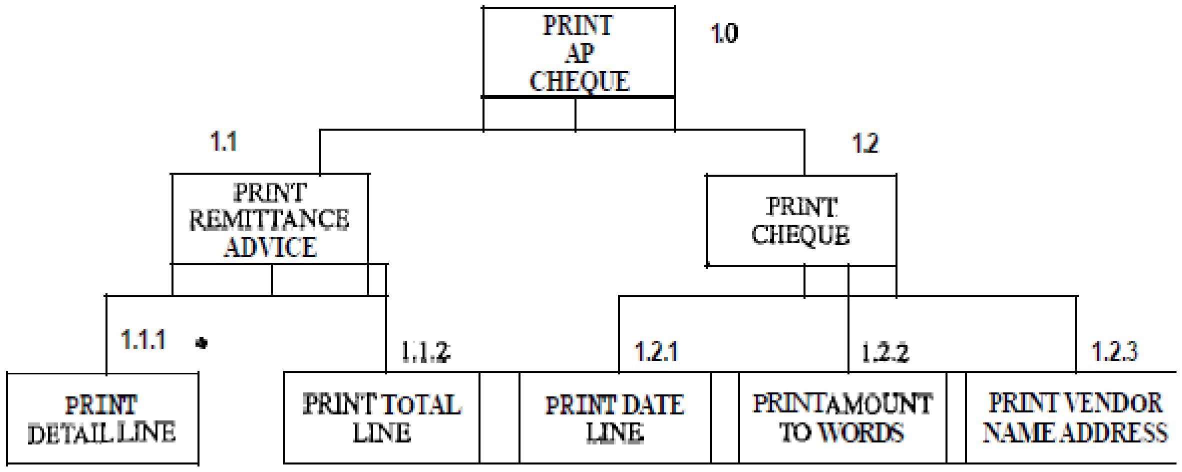


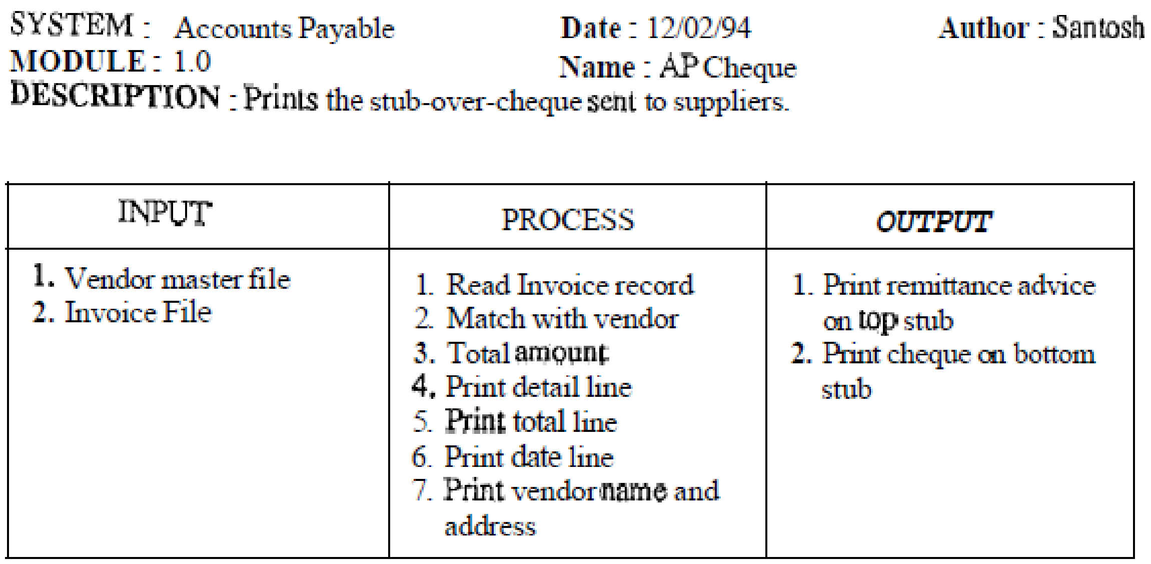
Fig. 6.15 Module development of the AP cheque system from the programmer's perspective

As far as the analyst is concerned, decomposition of the accounts payable cheque writing system can end at the third level of modules. However, the programmer who eventually receivesthe HIP0 chart probably would decompose the modules to even lower levels, thus establishing a new series of numbers (figure 6.15). Programmers sometimes call their VTOCs structure charts, hierarchy charts, or tree charts. When finally programming the accounts payable check system, the programmer would begin at the top left, programming each level before moving down and to the right. Thus theprogrammer first would code "remittance advice?' 1.1 followed by "print detail" (1.1.1) and then "print totals" (1.1.2). Having coded all of module 1.1, the programmer would tackle module 1.2, beginning with "print date line" (1.2.1), moving on to "print amount in words" (1.2.2), and finishing with "print vendor name address" (1.2.3).

Figure 6.15 shows three levels of modules, but complex systems may require many more.Regardless of their number, modulesshould receive unique and brief names that contain justenough detail for readers to understand their purposes.

6.4.2 Constructing an XPO

Let us add the second part of the HIPO diagramming system, the PO chart. Whereas the VTOC diagram graphically shows an overview of t hahe system, the IPO charts depict program logic, illustrating the steps required to produce desired output.



SYSTEM : Accounts Payable **Date :** 12/02/94 **Author :** Santosh
MODULE : 1.0 **Name :** AP Cheque
DESCRIPTION : Prints the stub-over-cheque sent to suppliers.

INPUT	PROCESS	*OUTPUT*
1. Vendor master file 2. Invoice File	1. Read Invoice record 2. Match with vendor 3. Total amount 4. Print detail line 5. Print total line 6. Print date line 7. Print vendor name and address	1. Print remittance advice on top stub 2. Print cheque on bottom stub

Fig. 6.16 The IPO chart detail or program logic

As figure 6.16 shows, the top of the PO chart identifies the module with its number, title, abrief description, date, and the analyst's name. The chart itself is divided into three units:data input (the names of the files used), processing activities eat will require programming,and output, which, in the case of our accounts payable system, would be the printed remittanceadvice (1.1) and the check (1.2). In the body of the chart, we use a narrative form todescribe the input, process, and output as a list of activities. It simply lists activities, notnecessarily ordering them in the sequence they should occur.

The HIPO system forms valuable system documentation and helps the analyst prepare reports as the system is being designed and developed these charts offer several advantages.First, they can be drawn or modified rapidly. Second, they allow the analystgraphically to convey the system to non computer people. Third, standardised symbols enablesome future analyst to grasp the system quickly. Finally, HIPO charts facilitate efficientschedules because they make it easy to estimate the time it will take to program a module,thus, simplifying programming assignments. The VTOC offers the analyst an alternative tothe system flowcharts whereas the IPO replaces the program or detail flowchart.

6.5 Decision Tables and Decision Trees

Decision tables and trees were developed long before the widespread use of computers. They not only isolate many conditions and possible actions, but they help ensure that nothing has been overlooked.

6.5.1 Decision Tables

The decision table is a chart with four sections listing all the logical conditions and actions. In addition the top section permits space for title, date, author, system and comment. The condition stub displays all the necessary tests or conditions. Like the diamond in a flowchart or the IF inpseudocode, these tests require yes or no answers. The condition stub always appears in the upper left-hand comer of the decision table, with each condition numberedto allow easy identification.

Thus,conditionstub is a list of all the necessary tests in a decision table. In the lower left-hand corner of the decision table we find the action stub where one may note all the processes desired in a given module. Actions, like conditions, receive numbers for identification -purposes. Thus,action Stub is a list of all the processes involved in a decision table.

The upper right comer provides space for the condition entry - all possible permutations ofyes and no responses related to the condition stub. The yes or no possibilities are arranged as a vertical column called rules. Rules are numbered 1, 2, 3, and so on. We can determine the number of rules in a decision table by the formula:

Number of rules = 2^N = 2^N where N represents the number of conditions and ^ means exponentiation. Thus, a decision table with four conditions has 16 (2^4 = 2x2x2x2=16) rulesone with six conditions has 64 rules and eight conditions yield 256 rules.

Thus, Condition entry is a list of allthe yes/no permutations in a decision table. The lowerright corner holds the action entry. X's or dots indicate whether an action should occur as aconsequence of the yes/no entries under condition entry. X's indication action; dots indicateno action.

Thus, Action entry indicates via dot or X whether something should happen in a decision table.

6.5.2 Decision Trees

A decision tree helps to show the paths that are possible in a design following an action or decision by the user. It provides an overview of the flow of control to be built into computer programs.Decision trees turn a decision table into a diagram. This tool is read from left to right, decisions result in a fork, and all branches end with an outcome. Trees can be easily read by nontechnical users who find decision tables too complex. Users readily grasp branches, forks, and outcomes.

They are a simple, but powerful form of multiple variable analysis. They provide unique capabilities to supplement, complement, and substitute for:

- Traditional statistical forms of analysis (such as multiple linear regressions)
- A variety of data mining tools and techniques (such as neural networks)
- Recently developed multidimensional forms of reporting and analysis found in the field of business intelligence. Decision trees are produced by algorithms that identify various ways of splitting a data set into branch-like segments. These segments form an inverted decision tree that originates with a root node at the top of the tree. The object of analysis is reflected in this root node as a simple, one-dimensional display in the decision tree interface. The name of the field of data that is the object of analysis is usually displayed, along with the spread or distribution of the values that are contained in that field.

Summary

- Requirements analysis is the first step in the system design process, where a user's requirements should be clarified and documented to generate the corresponding specifications.
- Graphical description of a system's data and how the processes transform the data is known as Data Flow Diagram (or DFD).
- An open-ended box represents a data/store data at rest, or a temporary repository of data.
- The line arrow is much more important in this method because it carries the data flow, the data into the transform and the data out of the transform.
- A data dictionary defines each term (called a data element) encountered during the analysis and design of a new system
- Aliases, or synonyms, are allowed when two or more entries show the same meaning; a vendor number may also be called a customer number.
- Data dictionaries seldom include information such as the number of records in, file, the frequency a process will run, or security factors such as passwords users must enter to gain access to sensitive data.
- Besides supporting documentation to one degree or another, the active data dictionary supports program and operations development by exporting database definitions and program data storage definitions for languages such as COBOL and Job Control Language (JCL) for execution-time performance.
- An in-line data dictionary is active during program execution, performing such feats as transaction validation and editing.
- The IBM data dictionary, which is actually a six linked databases, each with many segments, consists of standard categories and the infrastructure needed to "customise" installation categories.
- A decision tree helps to show the paths that are possible in a design following an action or decision by the user.

References

- *Introduction to Requirements Analysis and Specification* [Pdf] Available at: < http://www.site.uottawa.ca/~bochmann/SEG3101/Notes/SEG3101-ch3-1%20-%20Intro%20to%20Analysis%20and%20Specification.pdf > [Accessed 4 July 2013].
- *System Requirement Specifications (SRS)* [Pdf] Available at: < http://www.nyu.edu/classes/jcf/CSCI-GA.244001/handouts/Assignment1SampleSolution.pdf > [Accessed 4 July 2013].
- Wasson, C.S., 2006. *System Analysis, Design, and Development: Concepts, Principles, and Practices*, John Wiley & Sons, Inc., Hoboken, New Jersey.
- Dixit, J.B., 2007. *Structured System Analysis and Design*,Laxmi Publications, New Delhi.
- *Video 23 - The Software Requirements Specification* [Video online] Available at: <http://www.youtube.com/watch?v=_XTQjKhh6hQ> [Accessed 4 July 2013].
- *How to prepare a Software Requirement Specification (SRS)* [Video online] Available at: <http://www.youtube.com/watch?v=b8NvYoWbgjE> [Accessed 4 July 2013].

Recommended Reading

- Jalote, P., 1997. *An Integrated Approach to Software Engineering*, 2nd ed.,Springer-Verlag New York, Inc.
- Rajaraman, V.,*Analysis and Design of Information Systems*, 2nd ed.,PHI Learning Pvt. Ltd.
- Dennis, Wixom & Roth, D., 2009.*System Analysis And Design*, 3rd ed., John Wiley & Sons, Inc., Hoboken, New Jersey.

Self Assessment

1. Requirements analysis is the _______ step in the system design process, where a user's requirements should be clarified and documented to generate the corresponding specifications.
    a. first
    b. second
    c. fifth
    d. sixth

2. A ____________is a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristics of a system, and often, the procedures for determining whether these provisions have been satisfied.
    a. System
    b. Data
    c. Specification
    d. Software

3. A ________________ defines each term (called a data element) encountered during the analysis and design of a new system.
    a. logical structure design
    b. software
    c. data
    d. data dictionary

4. Aliases, or synonyms, are allowed when two or more entries show the same meaning; a vendor number may also be called a __________ number.
    a. data
    b. customer
    c. software
    d. components

5. Some commercial software packages, usually called _________________help the analysts maintain their dictionaries with the help of the computer
    a. DDS
    b. JCL
    c. software
    d. hardware

6. Which of the following statements is true?
    a. HIPO stands for Hierarchy plus Input Process Output.
    b. HIPO stands for High plus Input Process Output
    c. HIPO stands for Hollow plus Input Process Output
    d. HIPO stands for Height plus Input Process Output

7. Which of the following statements is false?
    a. Decomposition ends when all modules are double purpose, single entry; and single exit.
    b. An in-line data dictionary is active during program execution, performing such feats as transaction validation and editing.
    c. The IBM data dictionary, which is actually a six linked databases, each with many segments, consists of standard categories and the infrastructure needed to "customise" installation categories.
    d. A decision tree helps to show the paths that are possible in a design following an action or decision by the user.

8. An IPO chart defines the inputs, __________, and outputs for each module in the program.
    a. information
    b. processing
    c. software
    d. hardware

9. When used to diagram a program, the ________ arranges the program modules in order of priority, and it reads from the top down and from left to right
    a. HIPO
    b. Software
    c. VTOC
    d. Components

10. Decision trees are produced by ____________ that identify various ways of splitting a data set into branch-like segments.
    a. data
    b. software
    c. algorithms
    d. components

Chapter VII

System Control and Quality assurance

Aim

The aim of this chapter is to:

- introduce quality factor specification
- explain quality assurance in software life cycle
- discuss maintenance and support

Objectives

The objectives of this chapter are to:

- explore software testing and implementation
- elaborate maintenance and support issues
- define testing

Learning outcome

At the end of this chapter, you will be able to:

- distinguish levels of quality assurance
- understand designing test data
- analyse audit trail

7.1 Introduction

The amount and complexity of software developed today stagger the imagination. Software development strategies have not come up with the standard and because of this software products do not meet the application objectives. Subsequently control must be developed to ensure a quality product. Basically quality assurance is the review of software products and its related documentation for completeness, correctness, reliability and maintainability. System control and quality assurance deals with various issues involved in the quality assurance, testing and system control.

The objectives of system control and quality assurance are as follows:

- To highlight various issues involved in quality assurance, testing and system control.
- To point out the role of quality assurance in various stages of a Software Development Life Cycle.
- To illustrate the significance of various levels of tests and their different functions.
- To explain the role played by system control in a computer system.

7.2 Quality Assurance in Software Life Cycle

The software life cycle includes various stages of development and each stage has the goal of quality assurance. Steps taken in this regard are summarised below:

7.2.1 Quality Factors Specifications

The objective of this stage is to describe various factors which are responsible for quality of the anticipated system. The objectives are listed below.

- Correctness: The degree to which a program meets system specifications and user objectives falls under correctness.
- Reliability: The extent to which the system performs its intended functions over a time is known as reliability.
- Efficiency: Computer resources required by a program to perform a particular function.
- Usability: The efforts required to understand and operate a system.
- Maintainability: Maintainability is the ease with which the program errors are detected and removed.
- Testability: The efforts required to test a program to ensure its correct performance is one of the main objectives.
- Portability: The ease of transporting a program from one hardware configuration to another.
- Accuracy: The, required precision in input, editing, computations, and output should be accurate enough.
- Error tolerance: Error correction and detection versus error avoidance all together get accommodated in error tolerance.
- Expandability: Ease of expanding the existing database.
- Access control and audit: Control of access to the system and the extent to which that access can be audited is important.
- Communicativeness: Usefulness and effectiveness of the inputs and outputs of the system.

Various phases of specifications are requires in software life cycle, the phases are explained here below.

- Software Requirements Specifications: The quality assurance goal of this stage is to generate the requirements document that helps in providing technical specifications for developing the software.
- Software Design Specifications: In this stage, the software design document defines the overall architecture of the software that provides the functions and features given in the software requirements document.
- Software Testing and Implementation: The quality assurance goal of the testing phase is to ensure that completeness and accuracy of the system and minimize the retesting process. The goal of the implementation phase is to provide a logical order for the creation of the modules and also the formation of the system.

- Maintenance and Support: This phase provides the necessary software development for the system to continue to comply with the original specifications. The quality assurance objective is to develop a procedure for correcting errors and enhancing software.

7.3 Levels of Quality Assurance

Analysts use three levels of quality assurance: testing, verification with validation and certification.

7.3.1 Testing

System testing is quite expensive and time consuming process. The common view of testing held by users is that it is performed to prove that program is error free. But this is quite difficult since the analyst cannot prove that software is free from all sort or errors.

Therefore the most useful and practical approach is with the understanding that testing is the processing of executing a program with the explicit intention of error detection, which could be the reason for a program to fail. A successful test, at that time, is one that finds an error.

7.3.2 Verification with Validation

Verification is also intentional to find errors just like testing. Verification is performed by executing a program in a pretended environment. Validation refers to the process of using software in a live environment finding errors. Whenever the commercial systems are developed with the future aim of distributing them to dealers for sale purposes, they first go through verification; it is also called as alpha testing sometimes. The feedback from the validation phase generally brings some changes in the software to deal with errors and failures that are open. Then a set of user sites is selected for putting the system into use on a live basis. These beta test sites use the system in day-to-day activities; they process live transactions and produce normal system output. There is a possibility that the validation may continue for more than a few months. Throughout the course of validating the system, failure may occur at any point and the - software will be changed. Continued use may bring more failures and the need for still more changes.

7.3.3 Certification

The confirmation of a software package that it is developed with all the required standards is the last level of quality assurance and is known as certification. Importance of certification has increased because there is a growing demand for purchasing ready-to-use software in the market. After a package gets completely ready and certified it goes through a team of computer specialists. Computer specialists test, review and determine the software that how fine it achieves the user’s requirements and vendor’s claims. Certification is issued only if the package is successful in all the tests. Certification is important and a good policy but it does not mean that it is the best package to adopt. It is only the evidence of its performance what the vendor claims.

7.4 Design Objectives: Reliability and Maintenance

The two operational design objectives continually sought by developers are systems reliability and maintainability. This section will discuss about the importance of these objectives and ways to achieve them.

7.4.1 Designing Reliable Systems

A system is said to be reliable if it does not produce dangerous or costly failures during its normal use. The definition recognises that systems may not always be used according to the designer’s expectation. There are changes in the ways users use a system and also in business operations. However, there are steps analysts can follow to ensure that the system is reliable at the installation stage and its reliability will continue even after implementation.

There are two levels of reliability. The first level shows that the system is meeting the right requirements. This is possible only if a thorough and effective determination of systems requirements was performed by the analyst. A careful and thorough systems study is required for this aspect of reliability. The second level of systems reliability involves the actual working of the system delivered to the user. At this level, systems reliability is interwoven with software engineering and development.

Approach	Description	Example
Error avoidance	Prevents errors from occurring in the software.	It is impossible in large systems.
Error detection and correction	Recognises errors when they are encountered and corrects the error or the effect of the error so that system does not fail.	Traps and modifies illegal arithmetic steps Compensates for unexpected data values.
Error Tolerance	Recognises errors when they occur, but enables the system to keep running through degraded performance.	Shuts down part of the system. Does not perform some processing but keeps the system operational.

Table 7.1 Approaches to reliability

7.4.2 Designing Maintainable Systems

When the systems are installed, they are normally used for a considerable period. The average life of a system is generally 4 to 6 years, with the oldest applications often in use for over 10 years. However, this period of constant use brings with it the used to continually maintain the system. When system is fully implemented, analyst must take precautions to ensure that the need for maintenance is controlled through design and testing and the ability LO perform it is provided through proper design practices.

7.5 Maintenance Issues

Many studies at the private, University and government level have been conducted to learn about maintenance requirements for information systems. These studies reveal the following facts:

- From 60 to 90 per cent of the overall cost of software during the life of a system is spent on maintenance.
- Often maintenance is not done very efficiently.
- Software demand is growing at a faster rate than supply. Many programmers are spending more time on systems maintenance than on new software development.

Several studies of maintenance have examined the type of tasks performed under maintenance. Table 7.2 summarises the broad classes of maintenance found in information systems environments.

Category	Activity	Relative Frequency
Corrective	Emergency fixes, routine debugging.	20%
Adaptive	Accommodation of changes to data and files and to hardware and system software.	20%
Perfective	User enhancement, improved documentation, recording for computational efficiency.	60%

Table 7.2 Types of system maintenance

7.6 Maintainable Designs

The keys to reduce the need for maintenance, while making it possible to do essential tasks more proficiently, are listed below.

- User's requirements during systems development should be defined more accurately.
- Making better systems documentation.
- Using proper methods of designing processing logic and communicating it to project team members.
- Utilising the existing tools and techniques in an effective way.
- Managing the systems engineering processing a better and effective way.

Now it is clear that design is both a process and a product. The design practices followed for software has a great effect the maintainability of a system. Good design practices produce a product that can be maintained in a belter way.

7.7 Testing Practice and Plans

The philosophy behind testing is clearly to find errors. With this purpose in mind test cases are devised. A set of data processed by a system as normal input is the test case. However, the data are created with the express intent of determining whether the system will process them correctly. Test cases for inventory handling should include situations in which the quantities to be withdrawn from inventory exceed, equal and are less than the actual quantities on hand, this is one of the live examples of testing practice. Each test case is designed with the intention of finding errors in the way the system will process it.

There are two general strategies for testing software: Code testing and Specification testing. The analyst develops those cases to execute every instructions and path in a program in code testing. Whereas under specification testing, the analyst examines the program specifications and then writes test data to determine how the program operates under specific conditions. In spite of which strategy, the analyst follows; there are preferred practices to ensure that the testing is useful. The levels of tests and types of test data, combined with testing libraries, are important aspects of the actual test process.

7.8 Levels of Tests

It is necessary to perform both unit and system testing by the analyst because systems are not designed as entire systems nor are they tested as single systems.

7.8.1 Unit Testing

In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing focuses on the modules independently of one another, to find errors. This helps a tester in error detection in coding and logic that are contained within that module alone. The errors resulting from the communication between modules are initially avoided. Consider an example of a hotel information system; it consists of modules to handle reservations; guest checking and checkout; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing.

For each, it provides the ability to enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit testing should exercise each condition and option. Unit testing can be performed from the bottom up, starting will smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.

7.8.2 System Testing

The important and essential part of the system development phase, after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of errors in a newly designed system depend on some usual factors like communication between the user and the designer; the programmer's ability to generate a code that reflects exactly the system specifications and the time frame for the design sign.

Theoretically, a newly designed system should have all the parts or sub-systems are in working order, but in reality, each sub-system works independently. This is the time to gather all the subsystem into one pool and test the whole system to determine whether it meets the user requirements. This is the last change to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits. Testing is an important function to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

System testing consists of the following five steps:

Program testing

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. It is the responsibility of a programmer to have an error free program. At the time of testing the system, there exist two types of errors that should be checked. These errors are syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields out of range items, and invalid combinations. Since the logical errors are not detected by compiler, the programmer must examine the output carefully to detect them.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of the instructions, must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points.

String testing

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each part of the system is tested against the entire module with both test and live data before the whole system is ready to be tested.

System testing

System testing is designed to uncover weaknesses that were not found in earlier tests. This includes forced system failure and validation of total system as it will be implemented by its user in the operational environment. Under this testing, generally we take low volumes of transactions based on live data. This volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. When we see that the proposed system is successful in the test, the old system is discontinued.

System documentation

All design and test documentation should be well prepared and kept in the library for future reference. The library is the central location for maintenance of the new system.

User acceptance testing

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the system. After that a comprehensive test report is prepared. This report shows the system's tolerance, performance range, error rate and accuracy.

7.9 Special Systems Tests

There are other six tests which fall under special category. They are described below:

- Peak Load Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.
- Storage Testing: It determines the capacity of the system to store transaction data on a disk or in other files. For example, verify documentation statements that the system will store 10,000 records of 400 bytes length on a single flexible disk.
- Performance Time Testing: It determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.
- Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.
- Procedure testing: It determines the clarity of documentation on operation and use of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer,
- Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

7.10 Designing Test Data

The proper designing of test data is as important as the test itself. The test data as input are not valid or representation of the data to be provided by the user, then the reliability of the output is doubtful. Test data may be live or artificial. The live data is that which is actually extracted from the users' file. After a system is partially constructed, the programmers or analysts ask the users to key in a set of data from their normal activities. It is difficult to obtain live data in sufficient amount to conduct extensive testing.

The artificial test data is created solely for test purposes. Properly created artificial data should provide all combinations of values and formats and make it possible to test all logic and control paths through the program. Unlike live data, which are biased toward typical values, artificial data provide extreme values for testing the limits of the proposed system.

7.11 System Control

In a computer system, controls essentially mean the extent to which the system is secure against human errors, machine malfunction or deliberate mischief. The amount of control to be applied can vary depending upon the importance, critically and volume of the output. Depending on the nature of the system or the amounts of money at stake, the designer will need to build in different type of controls within system procedures, programs and operations.

7.11.1 Objective of System Control

Control mechanisms are designed to achieve the following objectives:

- Accuracy: The system should provide information and reports which are accurate in all respect.
- Reliability: The system should continue to function as it is designed to function. It should not break down due to malfunction of equipments.
- Security: Files and programs in the system must be secured against accidental damage or loss. Procedures must be established to restrict access to data by authorised user only.
- Efficiency: Efficiency of the system will essentially mean performing the tasks in the best manner and producing output of highest quality with least efforts.
- Audit: System controls should cover the aspect of auditing procedures. Facilities should be designed so that a transaction can be traced from its creation to its final use. This aspect of control assumes added significant in case of on-line systems where written documents may not exist for some transactions.
- Adherence to organisation policies: System controls should not conflict with organisation policies and must promote such policies. For example, by ensuring that payroll is produced accurately and by a specified time, the organisation objective of paying all employees in time will be promoted.

It may not always be possible to provide adequate controls in the system to meet the above objectives fully. Sometimes, it may happen that too many controls may adversely affect the performance of the system. On the other hand, lack of adequate control may create chaos and dissatisfaction. It is, therefore, a question of trade-off between the control and their objectives and as stated earlier, the decisions will depend on the nature of the system and criticality of its functions.

7.11.2 Types of Control

Two types of controls which affect the operation of a system are:

- External control
- Internal control

External controls to a system, as the name implies are laws, regulations, procedures and policies outside the scope of the system which affect the operations of the system. Internal controls are basically plans, procedures, guidelines, rules and checks under which the system must function. Much of these internal controls will be specified as a part of system design, but some of these may be internal controls of the organisation - with or without computerization.

7.12 Audit Trail

In on-line systems, unlike batch environments, there may not be copies of input source documents to fall back on if the system fails during processing. It is also possible for on-line users to sign on to a system, make changes in data already stored in files and sign of again without leaving a visible clue as to what happened. Unless the systems analyst develops an audit trail, no such protection exists in on-line systems.

Audit trail is the path which a transaction traces through a data processing system from source documents to summary reports. In other words, it refers to the facilities or procedures which allow a transaction to be traced through all stages of data processing beginning with its appearance on a source document and ending with its transformation into information on a final output document. The audit contains complete details such as reference numbers, dates, names which are recorded in files, ledgers and journals so that trailing of these records to source documents becomes easier. It is generally available in manual accounting system. But in computerised system, it is generally not available unless the system is specially designed to do so.

Audit trails are not primarily for the use of auditors. Rather they are quite important tools that are designed to help the management. The auditors use these tools which management has found necessary for internal purposes.

Summary

- The amount and complexity of software developed today stagger the imagination.
- Software development strategies have not come up with the standard and because of this software products do not meet the application objectives.
- The software life cycle includes various stages of development and each stage has the goal of quality assurance.
- The quality assurance goal of the testing phase is to ensure that completeness and accuracy of the system and minimize the retesting process.
- The goal of the implementation phase is to provide a logical order for the creation of the modules and also the formation of the system.
- System testing is quite expensive and time consuming process.
- Verification is also intentional to find errors just like testing.
- Validation refers to the process of using software in a live environment finding errors.
- The confirmation of a software package that it is developed with all the required standards is the last level of quality assurance and is known as certification.
- The two operational design objectives continually sought by developers are systems reliability and maintainability.
- A system is said to be reliable if it does not produce dangerous or costly failures during its normal use.
- The second level of systems reliability involves the actual working of the system delivered to the user.
- When the systems are installed, they are normally used for a considerable period. The average life of a system is generally 4 to 6 years, with the oldest applications often in use for over 10 years.
- The philosophy behind testing is clearly to find errors.
- In unit testing the analyst tests the programs making up a system.
- The important and essential part of the system development phase, after designing and developing the software is system testing.
- A program represents the logical elements of a system.
- When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of the instructions, must be traced to determine the problem.
- System testing is designed to uncover weaknesses that were not found in earlier tests.
- Peak load test determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand.

Reference

- Ainapure, B. S., 2009. *Software Testing And Quality Assurance*, Technical Publications.
- Puntambekar, A. A., 2010. *Software Engineering And Quality Assurance*, Technical Publications.
- *SOFTWARE TESTING* [Pdf] Available at: <http://elearning.vtu.ac.in/12/enotes/Soft_Test/Unit1-SV.pdf> [Accessed 4 July 2013].
- Chen, K. S., Chen, S. C. & Li, R. K., *Process Quality Analysis of Products* [Pdf] Available at: <http://ir.lib.ncut.edu.tw/bitstream/987654321/1712/1/21.Process%20Quality%20Analysis%20of%20Products.pdf> [Accessed 4 July 2013].
- *Learn Software Quality Life Cycle in Software Quality Assurance* [Video online] Available at: <http://www.youtube.com/watch?v=zv6-yPLxPlM> [Accessed 4 July 2013].
- *Software Testing Tutorial: Test Planning and Test Documentation – 1* [Video online] Available at: <http://www.youtube.com/watch?v=f3UmCgJZes0> [Accessed 4 July 2013].

Recommended Reading

- Patton, 2006. *Software Testing, 2/E*, Pearson Education India.
- Desikan, S. & Gopalaswamy, R., 2006. *Software Testing: Principles and Practices*, Pearson Education India.
- Burnstein, I., 2003. *Practical Software Testing: A Process-Oriented Approach*, Springer.

Self Assessment

1. Which of the following statements is false?
    a. Control must be developed to ensure a quality product.
    b. Quality assurance is the review of software products.
    c. System control and quality assurance do not explain the role played by system control in a computer system.
    d. The quality assurance goal of software requirements specifications is to generate the requirements document.

2. The degree to which a program meets system specifications and user objectives falls under ___________ factor.
    a. Correctness
    b. Reliability
    c. Efficiency
    d. Usability

3. Match the following

1. Usability	A. Usefulness and effectiveness of the inputs and outputs of the system.
2. Expandability	B. The ease of transporting a program from one hardware configuration to another.
3. Communicativeness	C. The efforts required to understand and operate a system.
4. Portability	D. Ease of expanding the existing database.

    a. 1-A, 2-D, 3-B, 4-C
    b. 1-B, 2-C, 3-A, 4-D
    c. 1-D, 2-A, 3-C, 4-B
    d. 1-C, 2-D, 3-A, 4-B

4. In which stage the software design document defines the overall architecture of the software that provides the functions and features given in the software requirements document?
    a. Software Requirements Specifications
    b. Software Design Specifications
    c. Software Testing and Implementation
    d. Maintenance and Support

5. __________ is performed by executing a program in a pretended environment.
    a. Verification
    b. Testing
    c. Certification
    d. Processing

6. When does a system called as a reliable system?
    a. A system is said to be reliable if two operational design objectives are continually sought by developers.
    b. A system is said to be reliable if second level of systems reliability involves the actual working of the system delivered to the user.
    c. A system is said to be reliable if it does not produce dangerous or costly failures during its normal use.
    d. A system is said to be reliable if its reliability is interwoven with software engineering and development.

7. ___________ focuses on the modules independently of one another, to find errors.
   a. Unit testing
   b. System Testing
   c. Program testing
   d. String Testing

8. __________ is designed to uncover weaknesses that were not found in earlier tests.
   a. String Testing
   b. Unit Testing
   c. Acceptance Testing
   d. System testing

9. Which of the followings is not an objective of system control?
   a. Security
   b. Audit
   c. Adherence to organisation policies
   d. Internal control

10. ________ is the path which a transaction traces through a data processing system from source documents to summary reports.
   a. External control
   b. Audit trail
   c. Peak load test
   d. Internal control

Chapter VIII

Introduction to Management Information System

Aim

The aim of this chapter is to:

- define management information system
- highlight the role of information in MIS
- describe various processes involved in MIS

Objectives

The objectives of this chapter are to:

- explain the concept of information and its role in management
- identify the concept of system and MIS
- elucidate information system process

Learning outcome

At the end of this chapter, you will be able to:

- understand the meaning and importance of MIS
- identify various processes involved in MIS
- describe computer based information system

8.1 Introduction

The capabilities, developments and achievements of human beings depend on their mind. Our mind gives us immense power or advantage, by using information as the raw material. Thus the well-being and success of any individual, enterprise or an economy is closely related to their ability. A project can accomplish significant development in its performance by ensuring that its managers are provided with appropriate information required by them for performing their work. Management Information System deals with understanding the ways and means of doing this effectively. It includes understanding two wide areas:

- One is concerned with the question of determining the nature of information required by the managers in an organisation and how organisation effectiveness can be improved by better information support.
- Second concern of MIS is developing ways of making the required information available effectively and economically. It requires understanding of information technology.

8.2 Importance of Information

Information has always played an essential and important role in working and development of human societies and the need and importance of information has been growing at an accelerated pace over ages. Information is needed for people to work cooperatively. No cooperative work is possible without the use of information to exchange knowledge and understanding on various issues like work to be performed collectively.

All the technological wonders of the 21st century are based upon the foundations of knowledge created over the period of ten thousand years. The most innovative technology or methods used today are the results of many small inventions, discoveries and improvements from the past. The technological developments in IT particularly the computers with an ability to accept, store and execute a series of user defined instructions enable organisations to use information more effectively.

8.3 Role of Information in Management

Information is essential for all the activities involved in our modern life. Latest technology cannot be used without information or understanding of the developments. Most of the activities of our present life are impossible without information. Managers need information about the environment of their organisation they manage and within which they operate. To work effectively they must understand the internal weaknesses and strengths of the organisation. They must be aware of the opportunities and external threats faced by the organisation.

Managers in organisations use and exchange information for different purposes. Coordination of activities of different persons and departments typically include information on the plan of work to be taken up, actual work done and/or progress made and actual results obtained.

Training and development activities implemented in an organisation to improve knowledge, skills and capabilities of an employee, need exchange of information. Policies, procedures, guidelines and similar communications intended to ensure some degree of control and uniformity in an organisation is carried out by delivering information from higher authorities.

- The effectiveness of a manager entirely depends on the nature and quality of information available and their ability to use the information.
- A successful manager identifies ways to develop, use and apply information received that can support the success of an organisation.

8.4 MIS and its Definition

Management Information System is mostly referred to as MIS which relates itself as collection of facilities and processes in an organisation that provides managers the information used by them.

Some of the definitions of MIS are
Henry C. Lucas
A set of organised procedures which, when executed, provides information to support decision-making.

Jayant K. Oke

An integrated system which transforms data (input) into reports (output) for facilitating decision-making through processing using various components of the information system vis., Hardware, Software, Database, Procedures, and personnel.

Kelly

A combination of human and computer based resources that result in collection, storage, retrieval, communication and use of data for the purpose of efficient management of operations and for business planning.

Krober Watson

An organised set of processes that provides information to managers to support the operation and decision making within an organisation.

T. Lucey

A system using formalised procedures to provide management at all levels in all functions with appropriate information, based on data from both internal and external sources, to enable them to make timely and effective decisions for planning, directing and controlling the activities for which they are responsible.

Milind Oka

An integrated system that consists of human and capital resources that enables collection, storage and processing of data and information to produce and communicate relevant information to different levels of management in order to provide requisite support in performing management related activities.

8.5 Systems Concept

- The systems concept and systems thinking can be used for understanding MIS better and for designing and developing effective and efficient MIS to suit the requirements of different organisations.
- A system is defined as a collection of interrelated parts forming a synergistic whole that jointly serves the desired purpose. The parts which form the whole system, also called components or elements of the system, can be things, people or both.
- The system as a whole receives inputs from sources outside the system and processes these inputs within the system. The product or results of these processes within the systems are then given out of the system as output of the system.
- A part of the output of a system may be fed back to it as input. This is called feedback. The purpose of feedback is to determine how a system is performing and guide action on improvement of system performance.
- These actions intended to improve system performance are called control actions.

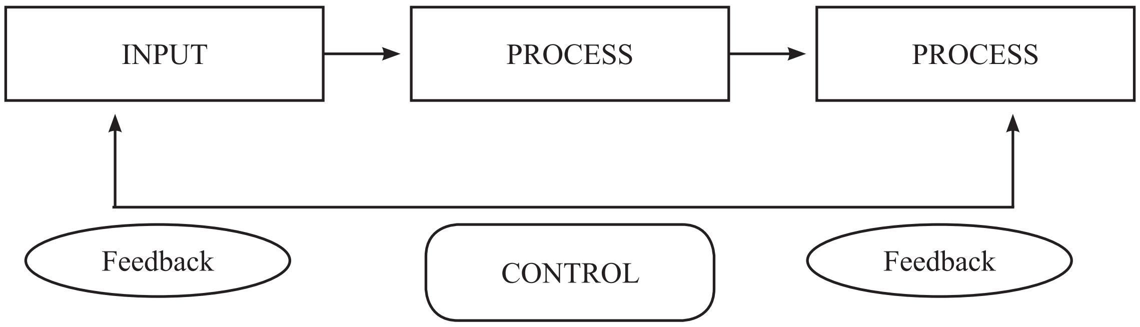


Fig. 8.1 Information system process

8.6 Characteristics of Useful Information

Various characteristics that determine the quality and suitability of information for the proposed use are:

- Relevant information is important for taking a decision. Information should be relevant to the purpose under consideration. Also having too much information is not required which leads to waste of efforts in sorting out relevant information from irrelevant.
- Many times it thus happens that information is collected but not used. Due to this, efforts are wasted and the user is distracted from more information they need.
- It is always recommended to exclude the data for which a clear use and user cannot be identified.
- Inaccurate information can cause more harm than good. Also too much insistence on accuracy can cause excessive delays and increase cost of information.
- An optimum balance between accuracy, cost and time must be maintained for precise in time delivery of information.
- Information has to be complete to be accurate. Incomplete information can be called correct but not accurate to the user's point of view.
- Reliability of information depends on the reliability of data collection.
- Reliability depends on the source of the information. Information must be gathered from reliable sources and rechecked with related sources.
- Information must be concise, as concise information covers only required facts. It covers only the information required by the user.
- Too much information causes information overload, where a decision maker has too much information and is unable to determine what is really important.
- Information becomes more meaningful to the user when it is appropriately analyzed.
- When people use information, they have to process mentally to make it a part of their internal knowledge. At times they may also process it externally to have a better understanding of the issues involved.
- The analysis and presentation must be decided also based on the user characteristics. The user must understand the information.
- Information must be made available when required. It is advantageous to have information available as early as possible.
- The value of information declines with time.
- Another significant consideration in deciding the nature of information is the degree to which information is secured from unauthorised access and modification.

8.7 Information System Process

Any information has three basic components – input, process and output. Both the input and output are different types of information, though the input information is called data in relation to the system under consideration to indicate that it has not been processed.

- Input to the system can also be the knowledge or understanding of individuals which gets transformed into information only when coded or represented in a form suitable for information processing.
- The various different types of processes that may take place within the system that accepts data within the system, converts it into information, and provides information to users.
- There are five main categories of information system processes:
  - Data Capture
  - Data Storage and Retrieval
  - Analysis
  - Information Presentation
  - Transmission

- Data capture involves coding of the input data in a form suitable for processing by the system. The most common form of manual data capture is a person observing something and describing one's observations orally or in writing.
- The captured data in the system may need to be stored for analyzing and presentation later. The most common method of storage is the paper record, stored as books and files.
- Analysis is the core of information processing, which transforms raw data into information.
- Presentation of information in a more easily understandable form is also one of the analytical tasks. Thus analysis also covers tasks like preparing graphs and reports.
- Presentation involves converting the information to a form understood by the user. Thus giving an oral or written report is two of the ways of information presentation used in manual systems.

8.8 Computer Based Information Systems

All MIS does not use computers, though most of the business organisations today use computer based information systems for their MIS.

- Computer based information systems are composed of hardware, software, telecommunications, people and procedures that are configured to collect, manipulate, store, and process data into information.
- Hardware is the equipment used to perform input, processing, and output devices. Software is the set of predefined instructions to the computer that determines the sequence of operations performed by the computer.
- Telecommunications allow organisations to link computer systems into effective networks. Networks can connect computers and computer equipment in a building, around the country, or across the world.
- Computer based information systems have enabled managers to automate some of the decisions that earlier required expert judgment of experienced managers.

8.9 Aims of Studying MIS

Following are the aims of studying MIS:

- Managers need to understand how information and information technology impact the working of any enterprise.
- They need to ensure that they and other managers reporting to them are getting good quality information required for effective management of the enterprise.
- They must understand how the nature of information available and the way it is used by managers impacts the performance of an enterprise.
- The aim of studying MIS is to develop the ability to design, install and operate a management information system that aims to:
- Make required information available for the management of an enterprise
- Improve the quality of information available to the managers
- Facilitate the best use of available information for improving managerial performance
- Reduce the total cost of getting and using management information

8.10 MIS and Operational Information

Organisations use information for carrying out operations as well as for managing them. Some examples of operations are placing purchase order on a supplier, determining the best way to machine a gear out of a blank, machining of such a gear, dispatching material from plant to the customer, and billing and payment collection at the check-out counter of a retail store.

All these operations require some information by way of operational data, and in turn may generate some additional operational data. The operational information system in a company is linked to its MIS in two ways.

- The operational system is the source of almost all internal data.
- Some of the external data form the initial input for the MIS.

8.11 Management Information System and Academics

The management is information system draws a lot of support from other academic disciplines too. The foundation of MIS is the management theory. It uses the principles and practices of management while designing the system, ant gives due regard to the theory of organisational behavior.

It considers the human mind as a processor of information. While designing the report format and forming communication channels, MIS takes into account the behavior of the manager as an individual and in a group. It gives due regard to the personal factors such as bias, thinking with a fixed frame of reference, risk aversion, strengths and weaknesses.

Another area of academics is operational research. The operational research is used for developing the models of management and they are then incorporated in the MIS as decision support systems. The inventory control, queuing theory, and resource programming are used in the MIS as decision support systems. The network theory is used for planning and controlling large projects. The application of PER / CPM to a project planning is now easily possible through the MIS support.

In the area of accounting application, it uses the accounting principles to ensure that the data is correct and valid. It uses the principles of double entry bookkeeping for balancing the accounts. It uses the accounting methodology for generating a trial balance sheet and other books of accounts.

The MIS uses the communication theory in a significant manner. The principle of feedback is used while designing analysis.. While designing the report format, attention is paid to avoid noise and distortions in the communication process.

The MIS further relies heavily on the decision methodology. It uses different mathematical techniques to handle the situation of decision making uses the method of decision- making under certainty for decision- making and action.

The MIS is based on database structures, viz., hierarchical; network and relational database have roots in the mathematics and the set theory. The MIS becomes rich in content and more useful when it becomes more and more a decision- making or decision- support system. The is possible when it builds decision making systems in MIS which in turn is possible if it draws tools, techniques, methods, rules and principles from pure and application science, and use them as an integral part of the system. The MIS draws data from its own source and uses it in the application of a variety of tools and techniques to solve the management, mathematics, and accounting.

8.12 MIS and the User

Every person in the organisation is a user of the MIS. The people in the organization operate at all levels in the hierarchy. A typical user is a clerk, an assistant, an officer, an executive or a manager. Each of them has a specific task and a role to play in the management of business. The MIS caters to the needs of all persons.

The main task of a clerk is to search the data, make a statement and submit it to the higher level. A clerk can use the MIS for a quick search and reporting the same to higher level. An assistant has the task of collecting and organising the data, and conducting a rudimentary analysis of integrating the data from different and disciplines to analyze it and make a critical comment if anything adverse is found.

The MIS offers the methods and facilities to integrate the data and report the same in a proper format. An executive plays the role of a decision maker. He is in of responsibility and accountability a position of a planner and a decision maker. He is responsible for achieving the target and goals of the organisation. The MIS provides facilities to analyze the data and offers the decision support systems to perform the task of execution. The MIS provides an action ñ oriented information.

The manager has a position of responsibility and accountability for the business results. His management role expands beyond his management function. He is a strategist and a long-term planner. He is a person with a foresight, an analytical ability and is expected to use these abilities in the functions of top management. The MIS provides information in a structured or unstructured format for him to react. The MIS caters to his constant changing needs of information. The user of the MIS is expected to be a rational person and the design of the MIS is based on this assumption.

However, in reality the impact created on individuals by MIS is difficult to explain. The nature of the impact in a few cases is negative. However, this negative impact can be handled with proper training and counseling.

It is observed that at lower level, is a sense of insecurity. As the MIS takes away the drudgery of search, collection, writing and reporting the data, the work vacuum, so created is not easily filled, thus creating a sense of insecurity. To some extent the importance of the person is also lost, giving rise to a fear of non-recognition in the organisation.

At the level of an officer and an executive, the MIS does the job of data manipulation and integration. It analyses the data in a predetermined manner. This means that the knowledge of business is transferred from an individual to the MIS and is made available to all in the organisation. This change arising out of the MIS creates a sense of being neglected for knowledge, information and advice. The psychological impact is larger if the person is not able to cope up with this change by expanding or enriching the job and the position held by him.

The manager holding a position in the top or middle management suffers from fear of challenge and exposure. The MIS makes these competitors more effective as they have access to the information and have an ability to interpret. This leads to a situation where he is afraid that that his position, decision and defense will be challenged and may be proved wrong sometime. The risk of adverse exposure to the higher management also increases. The effects so far pointed out are all negative and they are seen only in few cases.

The positive effects on the individuals at all levels are that they have become more effective operators. The time and energy which was spent earlier in unproductive work is now applied for a productive work. Some are able to use their analytical skills and knowledge with the information support for improving their position in the organisation. Managers, having improved their decision ñ making ability, are able to handle the complex situations with relative ease. Some are benefited by improving their performance and being held in high esteem by the higher management.

The enterprising managers are able to use the systems and the models for trying out a Number of alternatives in a given problem situation. The impact of the MIS on people of the organisation is phenomenal as it has made the same body of people collectively more effective and productive.

The recent major technological advances in communication such as Multimedia, Imaging. Graphical User Interfaces (GUI), Internet, Web etc. and the ability to access the data stored at different locations on the variety hardware of platforms would make MIS more attractive and efficient proposition. An intelligent user of information can demonstrate the ability of decision making, since his manipulative capability is considerably increased, with the information now being available on his desktop. Through the MIS, the information can be used as a strategic weapon to counter the threats to business, make business more competitive, and bring about the organisational transformation through integration. A good MIS also makes an organisation seamless by removing all the communication barriers

Summary

- In this chapter we have studied about information its importance and the part it plays in any organisation.
- The success of any individual, enterprise or an economy is closely related to their ability to get and use information effectively.
- Information system process follows three steps input, process and output. Data capture, data storage and retrieval, analysis, information presentation and transmission are the categories of information system processes.
- Computer based information system includes hardware, software, telecommunications and people from an organisation.

References

- Loudon, K. C. & Laudon, J. P., 2008. *Management Information Systems: Managing The Digital Firms*, Pearson Education India.
- Sadagopan, S., 2004. *Management Information Systems*, PHI Learning Pvt. Ltd.
- *Introduction to Management Information Systems*, [pdf] Available at: <http://www.mu.ac.in/mis.pdf> [Accessed 13 August 2012].
- *Management Information Systems*, [Online] Available at: <http://www.slideshare.net/bsetm/chapter-2-management-information-system-basics> [Accessed 13 August 2012].
- 2009. *Management Information Systems,* [Video Online] Available at: <http://www.youtube.com/watch?v=uTgSjnaL8Y8> [Accessed 13 August 2012].
- Watson, R., 2009. *Management Information Systems*, [Video Online] Available at: <http://www.youtube.com/watch?v=zYfZu5e5Qdk> [Accessed 13 August 2012].

Recommended Reading

- Murray, D., 2007. *Introduction to MIS*, Kendall Hunt Publication Co.
- Senn, J. A., 1989. *Analysis and Design of Information Systems*, Singapore, 2nd ed., McGraw-Hill Publishing Company.
- McLead, R., 1998. *Management Information Systems*, Prentice Hall.

Self Assessment

1. Information has always played an essential and important role in working and development of ___________.
   a. human societies
   b. organisation
   c. economy
   d. employees

2. ______________ activities implemented in an organisation to improve knowledge, skills and capabilities of an employee need exchange of information.
   a. Presentation
   b. Training and development
   c. Campaigning
   d. Forum

3. _____________information is important for taking a decision.
   a. Important
   b. Lengthy
   c. Relevant
   d. Precise

4. ________________ involves coding of the input data in a form suitable for processing by the system.
   a. Data Storage and Retrieval
   b. Analysis
   c. Data Capture
   d. Information Presentation

5. _______________ information can cause more harm than good.
   a. Important
   b. Lengthy
   c. Inaccurate
   d. Precise

6. Which of the following statements is true?
   a. MIS is developing ways of making the required information available effectively and economically.
   b. MIS cannot develop ways of making the required information available effectively and economically.
   c. MIS is developing ways of making the required information expensive.
   d. MIS is developing ways of making the required information available efficient.

7. Which of the following statements is true?
   a. All the technological wonders of the 21st century are based upon the knowledge crated over ten years.
   b. All the technological wonders of the 21st century are based upon the knowledge crated over thousand years.
   c. All the technological wonders of the 21st century are based upon the foundations of experience crated over period of ten thousand years.
   d. All the technological wonders of the 21st century are based upon the foundations of knowledge crated over period of ten thousand years.

8. Which of the following statements is true?
    a. Employees use information for carrying out operations as well as for managing them.
    b. Organisations use experiences for carrying out operations as well as for managing them.
    c. Organisations use information for carrying out process as well as develop them.
    d. Organisations use information for carrying out operations as well as for managing them.

9. Which of the following statements is true?
    a. It is always recommended to exclude the data for which a clear use and user cannot be identified.
    b. It is always recommended to include the data for which a clear use and user cannot be identified.
    c. It is always recommended to exclude the data for which a clear recipient cannot be identified.
    d. It is always recommended to exclude the information for which a clear use and user cannot be identified.

10. Which of the following statements is true?
    a. Reliability of information depends on the conciseness of data collection.
    b. Reliability of information depends on the reliability of data collection.
    c. Standard of information depends on the reliability of data collection.
    d. Reliability of information depends on the reliability of data execution.

Case Study I

E-Commerce site to enable members to buy and sell Portals/Domains

Requirement
In one of clients there was a need to System analysis study on a site, which can be a core for activities such as buying and selling of portals/domains by members. A robust but user-friendly site was what the customer desired.

Analysis
Asite of this concept would attract advanced and frequent visitors. User's sole intentions to benefit from the deals are to be assured by securing the site well. This should however not be a very restrictive site considering the growth potential.

Design
- Restrict Buy/Sell activities to members only.
- Take a Legally binding agreement signed by members who want to put up ads as well as Buy.
- Extensive usage of Client-side scripting to guide users at all point.
- Database incorporating best of RDBMS concepts with Referential integrity among various tables.
- Enable Session state and identify user at all stages and give a definite time to make page unavailable after that time.
- Use Site-Map to allow target browsing, by advanced users.
- Check the Domain names with an Admin and allow listing.
- Usage of E-Mail components to inform user of developments.

Conclusion
Customer was pleased with our analysis and design recommendations and went on to suggest our corporation for best of his in-house and other such jobs.

(Source:http://www.gantecusa.com/case9.asp)

Questions
1. What are the various requirements of an e-commerce site?

   Answers:

   In one of clients there was a need to System analysis study on a site, which can be a core for activities such as buying and selling of portals/domains by members. A robust but user-friendly site was what the customer desired.

2. What is the various users' sole intention?

   Answers:

   User's sole intentions to benefit from the deals are to be assured by securing the site well. This should however not be a very restrictive site considering the growth potential.

3. What was the final conclusion that came after this case study?

   Answers:

   Customer was pleased with our analysis and design recommendations and went on to suggest our corporation for best of his in-house and other such jobs.

Case Study II

System Analysis for Web Application, Click-A-Tutor

Requirement
The requirement is todevelop an educational web site which will cater to the needs of students in the age group of 6 –17. To make user-friendly interfaces and implement sturdy system to with stand errors by ignorant users.

Solution
The Gantec team started its work keeping the following key areas in focus.

- Understanding existing system of teaching and possible modification.
- Content Designing and allow Automatic Updating
- Selection of Web Technology and Application Development.
- Cost benefit analysis, for site owner and users as well

Design

- Develop and deploy database of MS SQL Server.
- Proper management of table spaces, Segments, Buffers and security options.
- Develop Administration tool using ASP technology for uploading and manipulating content in site.
- Studying & developing site module wise like registration, Content Management, Questions module, Admin and Tutor module
- Develop Credit card processing system.
- Modularity with the help of ActiveX Custom Component for data retrievals.
- Allow Index searching system
- A sturdy chat-server system to enhance interactivity among students and teachers.

Conclusion
This site was developed and is in the advanced stage of deployment. Client has vested great confidence observing our work and we are in the advanced stage to finalize contract for phase-II of the site.

(Source: http://www.gantecusa.com/case9.asp)

Questions

1. What are the requirements of the system analysis of the for the web applications?
2. What are the various solutions provided in this area?
3. Describe the design of the web analysis?

Case Study III

A Non-traditional Systems Analysis and Design

Introduction

Producing reliable, robust, cost-effective software systems has always been a problem forthe computing industry. In 1968, the term software crisis was coined by the NATO Software Engineering Conference for the myriad of problems in the development of quality software. In those days, computers were less common and more expensive; therewere few programmers and analysts; every system was developed from “scratch”. Evensmall projects took months to develop and were thus expensive undertakings.Computer Science (CS) and Computer Information Systems (CIS) programs haveresponded to the problem in a number of ways. Early programming courses taught notonly the particular language, but also how to develop quality programs; sometimes calledprogram engineering. Both programs provided courses in Systems Analysis and Designand later in Software Engineering.

Students were taught a systematic approach to the development of computer systems.They learned the lifecycle of a system and how to approach the analysis, design, andimplementation to produce reliable and robust software. In many programs, studentswere expected to demonstrate their competence through a project courses before theygraduated and joined an IT department and at the time, the only people developingsystems were the IT departments.

However, computing has changed since 1968. Computers are now cheap and plentiful.Modern programming languages have been developed, providing a number ofimprovements, most notably ease in developing a graphical user interface.A useful development, for creating software systems, has been the creation of applicationutilities – word processors, spreadsheets, data bases, etc. Now many “programming”solutions can be developed primarily by customising these packages.Furthermore, the combination of inexpensive hardware and application utilities hasallowed the end-user to develop their own unique software applications. Users no longerneed to wait for an IT department or consultant. Those who are interested may developtheir own. Development has moved from the hands of the CS/CIS, IT expert into thegeneral population.

The following is a study of one particular system developed in this new style: atechnology savvy user implementing a system with application utilities. The chosenproblem is simple and straight forward; one that could be approached by asophomore/junior CS/CIS major. Names and identities have not been used to ensureprivacy.

Problem

A local public school system provides special education services to preschool children bysending itinerant teachers to service these children at various preschool and day care facilities. These teachers must periodically (yearly) conduct “case conferences,” which provide progress reports using multiple state and local mandated forms. These multi-copy forms are preprinted, filled out by hand by the teacher, and then distributed to the parents and the student’s permanent file. The file is intended to follow the student as he/she enters the public school system.

The problem, one faced in many systems, is the massive paperwork. The itinerant teacher must fill out several forms for each student. These forms provide spaces for the teachers hand written information, however information is duplicated on the various forms and the process is time consuming. Current estimates are approximately one half hour per student with 60 or more students per teacher. In addition, the teachers must find time in their already busy schedule to complete the forms.

The teachers and administration felt this was an ideal situation for an improved system. The goals of the system were to be:

- reduced time to prepare the forms for a case conference
- reduce re-entry of redundant information
- provide more accurate records

Solution
The traditional approach would be to turn the problem over to the IT department or anoutside firm to analyze, design, and implement and appropriate solution. This would take time and be costly, and school systems rarely have extra funds.

The special education department had their own technology specialist. This person was trained as a special education teacher, but demonstrated an interest and aptitude in technology and was moved into an administrative position supporting technology in the special education area. This person had an advantage over a normal IT specialist: as a member of the department they were already familiar with the forms and the procedures.

Therefore, they initially had a better understanding of the problem than an IT designer. However, there was no more real analysis. The itinerant teachers were not consulted either as a group or individually; i.e. there was no discussion with the end-users. The manual system was taken as a model and replaced with a computerised version.

The solution implemented was to create files that were word processing (Microsoft Word) templates for each form that must be completed. These files were distributed to the teachers, on a floppy or compact disk. The individual teacher then, knowing the forms needed, would selected the appropriate files for a particular student, filled in the appropriate fields in the template, and saved the updated copies of the files in a machinereadable form, usually on floppy disk. The final step was the case conference where theitinerant teacher(s), and other teachers or specialists, met with the child's parents.

During the case conference, the teacher may need to add additional comments to the forms, so the files were again edited and then the appropriate number of hard copies was printed for signatures. The files were to be kept in machine readable form (floppy disk) for each student. Thus, a computerised record was formed for each student to be carried on through the educational process.

Analysis
The resulting system had several advantages over the previous manual system. Thematerial was no longer hand written. This meant no chance of going back to a form at a later date and finding that information was unreadable. The material was kept in a machine readable form, so in theory, it could be reused in subsequent years.

A change in the design of the forms could be implemented quickly, with minimal cost. The file containing the particular form could be edited, or replaced if there were major changes, and redistributed for the cost of a disk. The system was developed quickly and cheaply. The only personnel involved were already part of the staff. The system was implemented using existing equipment and software.

Unfortunately, there are a number of defects in the resulting system. The teachersinvolved were itinerant and many were not provided their own portable computers or printers. They were expected to find a computer at the facility they were serving or use the desk-top systems at their home office, which they visited only once or twice a week. This was a major problem when, during a case conference, updates were needed to the forms. The teacher had to find a computer to make the changes, print the updates, and then return to the conference. Many times a second meeting had to be scheduled to sign the updated forms.

There was no training planned or provided for the end-users, the itinerant teachers. Manyof the teachers had limited, or minimal, computer skills. They were expected to not only understand the working of the word processor, but also expected to be able to load a file from one location device, edit, then save it on another device. Furthermore, since they were itinerant, they may have to use several different computer systems. There was no part of the system designed to handle backing up the critical data stored on an unreliable floppy disk. There was no plan for error recovery.

The new system had three goals; the first was to reduce the time to prepare the forms. In fact, it took longer to create the forms for an individual student! The estimated time increased from half an hour for the hand written forms to about an hour for the word processing system. Typing speed accounts for a portion of the delay, but the new systemrequired opening multiple files and saving them on different media. While this is not a difficult task, it may be time consuming and stressful for a user who was uncomfortable with computers to begin with. Furthermore, there was no training to do this and no systematic approach to renaming and organising the new files.

The second goal was to reduce the re-entry of redundant data. This problem was not even addressed. The computerised forms were identical to the paper forms and therefore required all the re-entry the manual system required. The teachers had to re-enter data - such as name, birth data, etc. – repeatedly on the separate forms. This presented the same problems that the manual hand written system had. Furthermore, in the manual system the teacher could lay out the forms and copy the redundant information from one form to the others, thus ensuring some consistency.

In the computerised system, teacher workedon one form at a time and except for the most technologically savvy, could not compare the redundant data for consistency. The third goal was more accurate records. It does not appear that the computerised system provided any advantage over the manual system, although it appears to be no worse than the manual system.

As a final observation the newer system should have taken advantage of the potential toreduce data entry errors. The new system by default took advantage of the word processor's spell checking. However, additional information could have been checked. Dates and ages could have been checked for reasonableness and consistency. Scores on standardised tests could be checked for validity and consistency.

Conclusions
It is fairly obvious that the designer in this example did not follow a traditional analysis and design approach. That, in itself, does not mean the approach is wrong. It illustrates the way many systems are being developed. Even well-planned, well-staffed projects can fail.

Whether we, as computer professionals, like it or not, this is the wave of the future. Salespeople are developing their own customer databases. Doctors are setting up their own networks and developing customised applications. Any technologically savvy individual can now create a new system. When such a system is used only by the individual, then any effects of the system only involve that individual. However, if the system affects others, directly or indirectly, then we may hold the system to a higher standard. We don't want inaccurate patient records in a doctor's office and we don't want inaccurate records in student records.

In the case study, the designer was a knowledgeable user and therefore understood theproblem area. The designer didn't need to consult the users to understand how the forms were being created and used. However, this also meant that the other users had no input into the design process; they had no chance to voice concerns or evaluate proposed solutions.

In the case study, the designer used the software product (word processing) that theyknew best. While any designer does that; most professional systems designers have had a reasonably wide set of training and experiences. This designer was comfortable with word processing and little else. Thus, they didn't explore all the options available. A simple change to using a database package, like Microsoft Access, could have produced a much more useful system; one in which the teachers use one standard "form" for input of all relevant information and then produce the necessary reports. However, if the designer has had no experience with a database package then they have no way of knowingthat the option exists.

As noted before, the resulting system suffered from a number of other deficiencies. There was no method of error handling; there were no procedures for backups and failure; in fact, there were no procedures at all; the system provided no attempt at data consistency. We are entering a new software crisis. The original crisis was concerned with the development, by professionals, of reliable and robust software in a cost-effective manner. The new crisis does not deal with cost, since the investment in a computer system is minimal. Rather, the crisis is with the production of reliable, robust software systems by anyone. More and more software, like the example case study, is being created by individuals whose only credentials are an interest in technology.

(Source: http://www.micsymposium.org/mics_2005/papers/paper114.pdf)

Questions

1. What were the major problems in non-traditional systems analysis and design?
2. What were the goals of the system?
3. Who coined the term software crisisin 1968?

Bibliography

References

- Ainapure, B. S., 2009. *Software Testing and Quality Assurance*, Technical Publications.
- *An Introduction to Structured System Analysis and Design Method* [Pdf] Available at: < http://www.shere2000.co.uk/pdf/ssadm01.pdf> [Accessed 25 June 2013].
- Bellgran, M., Sèafsten, K., 2010. *Production Development: Design and Operation of Production Systems*, Springer.
- Chen, K. S., Chen, S. C. & Li, R. K., *Process Quality Analysis of Products* [Pdf] Available at: <http://ir.lib.ncut.edu.tw/bitstream/987654321/1712/1/21.Process%20Quality%20Analysis%20of%20Products.pdf> [Accessed 4 July 2013].
- *Computer Technologies* [Video online] Available at: <http://www.youtube.com/watch?v=X6tPgXPNR3c> [Accessed 25 June 2013].
- Dixit, J.B., 2007. *Structured System Analysis and Design*, Firewall Media.
- Dixit, J.B., 2007. *Structured System Analysis and Design*, Laxmi Publications, New Delhi.
- Downs, E. & Peter, C., 1992. *Structured Systems Analysis and Design Method*: *Application and Context*, Prentice Hall.
- *File Documentation* [Pdf] Available at: < http://foi.nuim.ie/documents/filing_guidelines.pdf> [Accessed 28 June 2013].
- *GNOME Documentation System* [Pdf] Available at: < https://developer.gnome.org/gdp-handbook/stable/gnomedocsystem.html.en> [Accessed 28 June 2013].
- Gupta, R.K., 2004. *Formal Methods and Models for System Design*: *A System Level Perspective*, Springer.
- *How to prepare a Software Requirement Specification (SRS)* [Video online] Available at: <http://www.youtube.com/watch?v=b8NvYoWbgjE> [Accessed 4 July 2013].
- *Information System Analysis and Design - Modeling System Requirements* [Video online] Available at: < http://www.youtube.com/watch?v=urI2aH5fqX0> [Accessed 1 July 2013].
- *Introduction to Document Management Software by Docsvault* [Video online] Available at: <http://www.youtube.com/watch?v=Cf-tKLgYwMo > [Accessed 28 June 2013].
- *Introduction to Management Information Systems*, [Pdf] Available at: <http://www.mu.ac.in/mis.pdf> [Accessed 13 August 2012].
- *Introduction to Requirements Analysis and Specification* [Pdf] Available at: < http://www.site.uottawa.ca/~bochmann/SEG3101/Notes/SEG3101-ch3-1%20-%20Intro%20to%20Analysis%20and%20Specification.pdf > [Accessed 4 July 2013].
- John, W. S., Robert, B. J., Stephen, D. B., 2011. *Systems Analysis and Design in a Changing World*, Cengage Learning.
- *Learn Software Quality Life Cycle in Software Quality Assurance* [Video online] Available at: <http://www.youtube.com/watch?v=zv6-yPLxPlM> [Accessed 4 July 2013].
- Loudon, K. C. & Laudon, J. P., 2008. *Management Information Systems: Managing the Digital Firms*, Pearson Education India.
- *Management Information Systems*, [Online] Available at: <http://www.slideshare.net/bsetm/chapter-2-management-information-system-basics> [Accessed 13 August 2012].
- *Management Information Systems*, [Video online] Available at: <http://www.youtube.com/watch?v=uTgSjnaL8Y8> [Accessed 13 August 2012].
- *Management Information Systems,* [Video online] Available at: <http://www.youtube.com/watch?v=zYfZu5e5Qdk> [Accessed 13 August 2012].

- Meister, D., Thomas P. E., 2001. *Human Factors in System Design, Development, and Testing*, Psychology Press.
- *MIS- Business Process and Information System.mp4* [Video online] Available at: <http://www.youtube.com/watch?v=ikNlsqD0bME > [Accessed 25 June 2013].
- *Modern System Analysis and Design* [Video online] Available at: <http://www.youtube.com/watch?v=fxzPolwXGL8> [Accessed 25 June 2013].
- Penny, A. K., 1992. *Introduction to systems analysis and design: a structured approach*, Wm. C. Brown Publishers.
- *Process Modeling for Information System Design* [Pdf] Available at: < http://courses.ischool.berkeley.edu/i290-1/f08/lectures/ISSD-20081020.pdf> [Accessed 1 July 2013].
- Prof. Desai, H.K., Gayatri, S.K., 2008. *Mcs-014 Systems Analysis and Design*, Dotcom Publications.
- Puntambekar, A. A., 2010. *Software Engineering and Quality Assurance*, Technical Publications.
- Ravichandran, D., 2001. *Introduction to Computers and Communication*, Tata McGraw-Hill Education.
- Sadagopan, S., 2004. *Management Information Systems*, PHI Learning Pvt. Ltd.
- *SLPSoft Software System Design and Modeling Version 2013.2* [Video online] Available at: < http://www.youtube.com/watch?v=XuJ3GcMvhHQ> [Accessed 1 July 2013].
- *Software Testing* [Pdf] Available at: <http://elearning.vtu.ac.in/12/enotes/Soft_Test/Unit1-SV.pdf> [Accessed 4 July 2013].
- *Software Testing Tutorial: Test Planning and Test Documentation – 1* [Video online] Available at: <http://www.youtube.com/watch?v=f3UmCgJZes0> [Accessed 4 July 2013].
- *Structured System Analysis and Design Method* [Pdf] Available at: <http://www.imse.hku.hk/imse2013/ie2013-31.pdf > [Accessed 2 July 2013].
- *Structured System Design* [Pdf] Available at: < http://www.thedirectdata.com/materials/ce/se/3%20Structured%20System%20Design.pdf> [Accessed 2 July 2013].
- *Superior Office Systems Document Management Introduction.mp4* [Video online] Available at: <http://www.youtube.com/watch?v=inFBEnt-ppk> [Accessed 28 June 2013].
- *System Analysis* [Pdf] Available at: < http://newton.uor.edu/courses/sysanades/pdf/anaintro.pdf> [Accessed 25 June 2013].
- *System Analysis and Design Lecture 1(Burraq Academy).flv* [Video online] Available at: <http://www.youtube.com/watch?v=F7-C7ci6ojo > [Accessed 2 July 2013].
- *System Development Lifecycle* [Pdf] Available at: < http://www.security.mtu.edu/policies-procedures/SystemDevelopmentLifecycle.pdf> [Accessed 25 June 2013].
- *System models* [Pdf] Available at: <http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch8.pdf > [Accessed 1 July 2013].
- *System Requirement Specifications (SRS)* [Pdf] Available at: < http://www.nyu.edu/classes/jcf/CSCI-GA.244001/handouts/Assignment1SampleSolution.pdf > [Accessed 4 July 2013].
- *Systems Development Life-Cycle Policy* [Pdf] Available at: < http://www.house.gov/content/cao/procurement/ref-docs/SDLCPOL.pdf> [Accessed 25 June 2013].
- *Video 23 - The Software Requirements Specification* [Video online] Available at: <http://www.youtube.com/watch?v=_XTQjKhh6hQ> [Accessed 4 July 2013].
- Wasson, C.S., 2006. *System Analysis, Design, and Development: Concepts, Principles, and Practices*, John Wiley & Sons, Inc., Hoboken, New Jersey.
- *What is Structured Cabling Standard* (TIA-568-C)? [Video online] Available at: <http://www.youtube.com/watch?v=NRE6O_mvFus> [Accessed 2 July 2013].
- Yildirim, H., & Osita, D.I., 2001. *The Mechanical Systems Design Handbook: Modeling, Measurement, and Control*, CRC Press.

Recommended Reading

- Burnstein, I., 2003. *Practical Software Testing: A Process-Oriented Approach*, Springer.
- Dennis, M. B., 2011. *The Engineering Design of Systems: Models and Methods*, John Wiley & Sons.
- Dennis, Wixom & Roth, D., 2009.*System Analysis and Design*, 3rd ed., John Wiley & Sons, Inc., Hoboken, New Jersey.
- Desikan, S. & Gopalaswamy, R., 2006. *Software Testing: Principles and Practices*, Pearson Education India.
- Edward, Y., Larry, L.C., 1997. *Structured design: fundamentals of a discipline of computer program and systems design*, Prentice Hall.
- Gupta, P., 2008. *System Analysis and Design*, Firewall Media.
- Harry, K., 1976. *Systems Design End Documentation: An Introduction to the HIPO Method*, Van Nostrand Reinhold Company.
- Jalote, P., 1997. *An Integrated Approach to Software Engineering*, 2nd ed., Springer-Verlag New York, Inc.
- Jean, M.B. & Jacques, R., 1997. *Models in System Design*, Springer.
- Jeffrey, L. W., Lonnie, D. B., 2008. *Introduction to Systems Analysis and Design*, McGraw Hill Irwin.
- Jeffrey, O. G., 2010. *System Requirements Analysis*, Academic Press.
- Kaujalgi, V.B., 1994. *Structured Systems Analysis and Design: Data Flow Approach*, Orient Blackswan.
- Kelkar, S.A., 2004. *Structured Systems Analysis and Design*, PHI Learning Pvt. Ltd.
- McLead, R., 1998. *Management Information Systems*, Prentice Hall.
- Murray, D., 2007. *Introduction to MIS*, Kendall Hunt Publication Co.
- Naveen, P., Colette, R., Barbara, P., 1993. *Information system development process: proceedings of the IFIP WG8.1 Working Conference on Information Development Process*, North-Holland.
- Ned, K., 2006. *Systems Analysis & Design Fundamentals: A Business Process Redesign Approach*, SAGE.
- O'Brien, J.A., 2004. *Introduction to Information Systems*, Tata McGraw-Hill Education.
- Patton, 2006. *Software Testing, 2/E*, Pearson Education India.
- Rajaraman, V., *Analysis and Design of Information Systems*, 2nd ed., PHI Learning Pvt. Ltd.
- Renevan, L., Gilles, S., 2011. *Integrated Circuit and System Design*. *Power and Timing Modeling, Optimization, and Simulation: 20th International Workshop*, Springer.
- Richard, W. P., Anne, S.M., 2007. *Human-System Integration in the System Development Process: A New Look*, National Academies Press.
- Senn, J. A., 1989. *Analysis and Design of Information Systems, Singapore, 2nd ed.*, McGraw-Hill Publishing Company.
- William, B.G., 1993. *Introduction to Electronic Document Management Systems*, Academic Press.

Self Assessment Answers

Chapter I

1. a
2. c
3. d
4. b
5. a
6. a
7. d
8. b
9. a
10. c

Chapter II

1. b
2. a
3. c
4. d
5. a
6. c
7. a
8. a
9. b
10. a

Chapter III

1. a
2. a
3. d
4. c
5. d
6. a
7. c
8. c
9. a
10. c

Chapter IV

1. d
2. b
3. a
4. b
5. d
6. c
7. a
8. b
9. a
10. d

Chapter IV

1. a
2. c
3. d
4. a
5. d
6. b
7. a
8. b
9. c
10. a

Chapter VI

1. a
2. c
3. d
4. b
5. a
6. a
7. a
8. b
9. c
10. c

Chapter VII

1. c
2. a
3. d
4. b
5. a
6. c
7. a
8. d
9. d
10. b

Chapter VIII

1. a
2. b
3. c
4. c
5. c
6. a
7. d
8. d
9. a
10. b