UNIT -II

Designing (Technical, Detailed, High Level): Introduction to Technical Design and Construction. A Client Server Model of E-Commerce, Understanding Technical Design, Construction. Introduction to Detail Design, High-level Design, Performing High-Level Design, High Level design of Business transactions, Applying' High-Level design with example.

11

Technical Design

11.1 Introduction

Next step to be followed in the Web E-process is the engineering step. This step is basically the collection of two sub steps: Non-Technical Design, and Technical Design, which are performed in parallel to each other. The goals defined in these two sub steps are collectively taken as the complete goal set of engineering step. Now, we will discuss these two tasks one-by-one in sequence.

Technical design has a high number of applications and is a very popular occupation. A degree in technical design will prepare someone for a job as a CNC operator, a CAD designer, drafters and even prototype designers. Drafters draw up plans that are used to create many of the items we use every day, including the microchips the operate computers, microwaves or even televisions. Technical design is one of the factors that advances our technology. The new slimmer cell phones, the next wave in flat screen televisions or the new fuel-efficient car are all results of technical design. The designs are normally drawn up on CAD or computer aided design system to make sure that the measurements are all exact. Mechanical ability and visual aptitude are important factors for anyone that is interested in moving into this field of employment. Artistic ability can be helpful, but most of the programs used for design can help make up any lack in that area. It is also important for someone in this field to have good interpersonal skills due to people in technical design interacting and working with so many different people in various fields.

11.1.1 Non-Technical Design

The first step performed in this step sequence of engineering activity is the non-technical design. This design activity is performed by the non-technical members of the Web E-team. This step also consists of two types of designs : Content Design, and Production, to be performed. They are discussed below.

A. Content Design

The designing of contents is done by the non-technical members pf WebE team/ These non-

technical members may be copywriters, artists, graphic designers, and others who generate Web-

based contents.

Goal : During Content designing, following goals are required to be achieved:

- Overall structure and detailed layout of the information content is derived that will be presented as part of the' WebApp.
- Text, graphics, images, audio and video contents are designed.
- Design models are prepared.

Description : The mentioned goals of the content design step can be easily performed by the non-technical members of the Web E-team because, here, they have to design only various types of contents of the WebApp, and, for this purpose, they can use some automated tools, methods, procedures, and standards, if required. Additionally, if they have experience and perfection in their work, they can represent the contents and information of WebApp in a very attractive and systematic manner.

B. Production

Production task is the second task to be performed in the non-technical design activity. This task is also performed by the non-technical members of the Web E-team.

Goal : The main goal to be performed during this step is to produce the text, graphical images, audio and video contents.

Description : Once the content designing is completed, all the contents and information of WebApp are developed and produced by Web E-team. As the designs of these are developed during above step, so, all these contents are produced and generated during this step by the non-technical members only.

In parallel to the progression of these steps A and B. another set of steps is also performed, which is discussed next

11.1.2 Technical Design

Second step performed in parallel of non-technical design in the engineering activity is the technical design work which is performed by the technical members of the Web E-team. The technical members of the Web E-team can be Web engineers and developers only. During technical design step, the structure of the WebApp is established and also the user WebApp interaction flow is checked.

Technical design is logical design made on a specific softwar system.Technical design is used in many fields and applications, including drafting and construction. This skill is very important in modern manufacturing applications

In order to perform technical design activity effectively there are four technical elements which a Web engineer should work to reuse. These technica elements are discussed below.

1. Design Principles and Methods

Various design principles and concepts which are applied in software engineering design can be properly used for developing WebApps. So, the design fundamentals, like decomposition and modularity, abstraction, stepwise refinement, software architecture, control hierarchy, data structure, software procedure, structural partitioning and information hiding can be used easily and without any doubt. In addition to these, design methods for object-oriented systems and diagrammatic notations of UML can also be adapted for use in design activities for WebApps.

2. Golden Rules

A variety of golden rules or design heuristics can be used during the design of WebApp which we apply in software engineering products.

3. Design Patterns

Various types of design patterns can be adopted for developing WebApps. This generic approach is used for solving some small problems that can be adapted to a much wider variety of specific problems. A variety of design patterns are discussed later.

4. Templates

A template can be used to provide us a skeletal layout for any design pattern or document that is to be used within the WebApp. According to Nanard and Kahn, a template is defined as :

"Once a template & defined, any part of a hypermedia structure that conforms to this template can be automatically generated or updated just by calling the template with relevant data. The use of constructive templates implicitly relies on the separation of hypermedia document contents from the specification of its presentation: source data are mapped into the hypertext structure as specified in the template."

After discussing these four essential elements of design, we would now turn our attention to three important and sequential steps : Architecture Design, Navigation Design, and Interface Design, which are performed during technical design step. They are described below one-by-one.

(A) Architectural Design

Architectural design is the first step performed in technical design step, which basically concerns with the architecture of the WebApp.

Goal : The goals to be achieved during this step are :

- The structure of the overall hypermedia of WebApp is defined.
- Design patterns are defined and described.
- Constructive templates are defined to develop the structure and to achieve reusability.

Description : To achieve these goals of the architectural design, two aspects are essentially required to be defined which can help in developing the structure of the WebApp. These two aspects are :

1. WebApp Structures
2. Design Patterns

1. WebApp Structures

To develop a WebApp, a WebApp architectural structure is required. As such, the WebApp architectural structures and layouts depend on the following factors :

- goals established for a WebApp,
- the contents to be presented,
- the users who will visit this WebApp, and,
- the navigation philosophy.

Based on these aspects, any of the architectural layout can be chosen if we have any choice. In this direction, Powelt has suggested four types of architectural structures, given as :

A. Linear Structures
B. Grid Structures
C. Hierarchical Structures
D. Networked or Pure Web Structures

A. Linear Structures

Linear structures are the structures in which Web pages are linearly connected or related to each other. These Web pages are associated with each other in a sequence. These linear structures are also categorized in three types as:

a. Simple Linear : In such type of structures, Web pages have only a single linear sequence as shown in fig. 11.1.

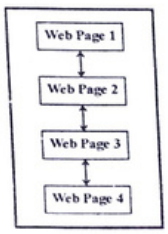

Fig. 11.1 : Linear Structure – Simple Linear

b. Linear with Optional Flow : In such type of structures, a linearly defined sequence is followed but some options are also included at some places. Such type of structure is shown diagrammatically in fig. 11.2

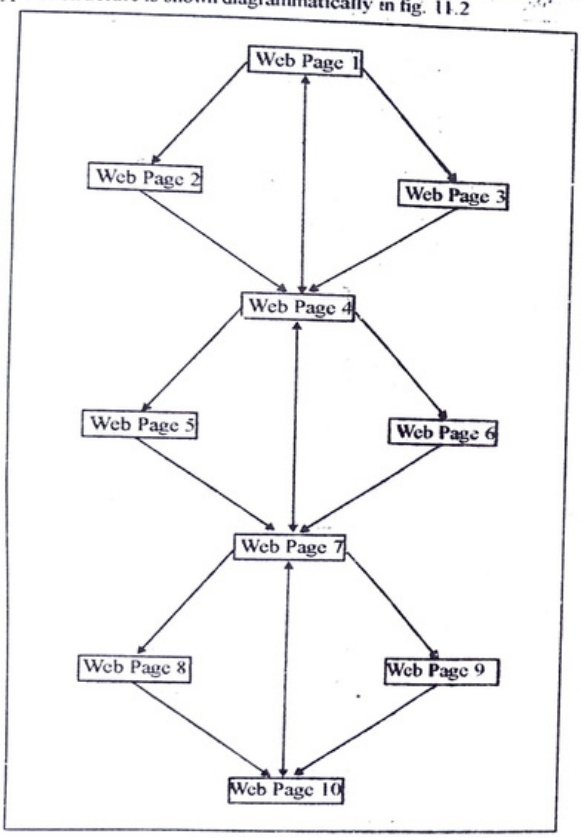

Fig 11.2 : Linear Structure – Linear with Optional Flow

c. Linear with Diversions : These type of structures are more complex ones than the previous ones. In these type of structures, some diversions are also included among the Web pages. Such type of structure is illustrated in fig. 11.3.

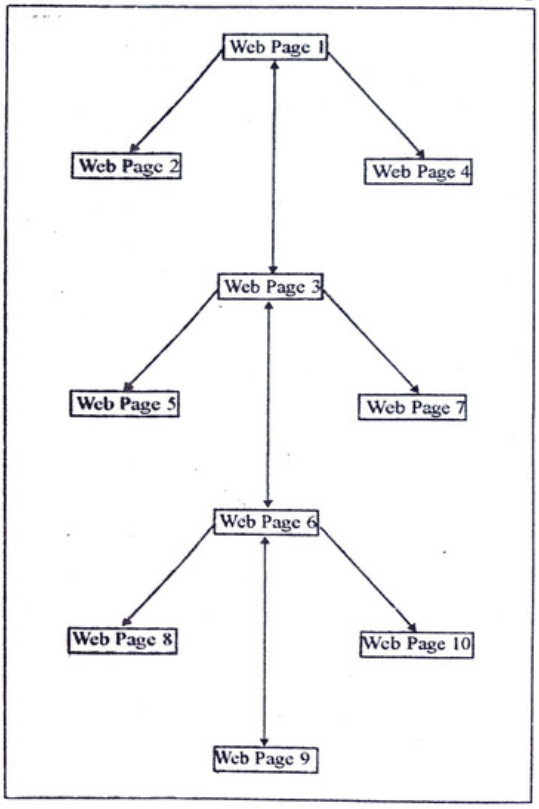

Fig. 11.3 : Linear Structure – Linear with Diversions

B. Grid Structures

Grid structures are an architectural category that can be applied when WebApp content can be organized categorically in two (or more dimensions). If two – dimensional grid structure is considered then horizontal and vertical dimensions are taken to build up this structure. This grid structure is shown in fig. 11.4.

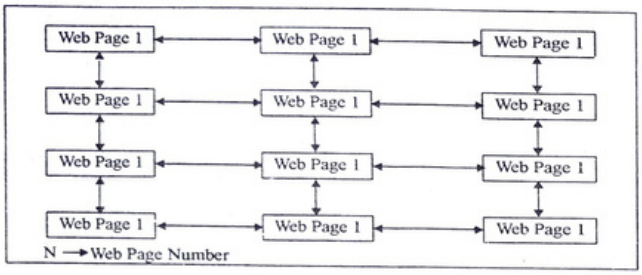

Fig. 11.4 : Grid Structure

C. Hierarchical Structures

Hierarchical structures are also called as the tree structures and are the most commonly used structures for WebApps. A hierarchical structure is designed in a manner that enables flow of control horizontally, across vertical branches of the structure. That's why, contents presented on the far left-hand branch of the hierarchy can have hypertext links that lead to content that exists in the middle or right-hand branch of the structure. This type of structure is shown in fig. 11.5.

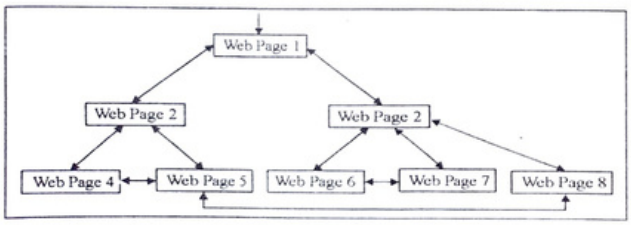

Fig. 11.5 : Hierarchical Structure

D. Networked of Pure Web Structures

In such type of structures, architectural components or Web pages are designed in a manner so that they may pass control (via hypertext links) to virtually every other Web page in the system. Its structure is illustrated in fig. 11.6.

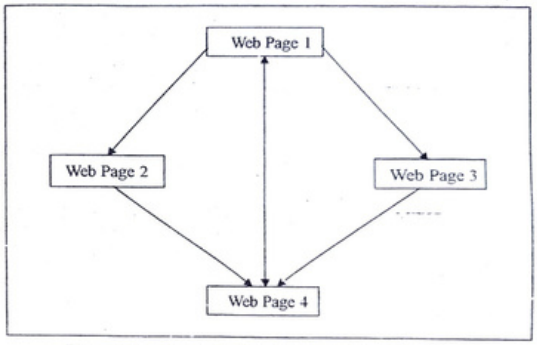

Fig. 11.6 : Networked or Pure Web Structures

2. Design Patterns

As we have discussed earlier that design patterns are a generic approach for solving some small problem that can be adapted to a much wider variety of specific problems. So, in WebApps, design patterns can be applied at three levels – Architectural, Component and Hypertext.

The architectural and component level design patterns are used for the data processing functionality of the WebApp, whereas the hypertext – level design patterns are used for navigation features when a user move through WebApp contents. Bernstein has provided a list of design patterns, which are shown with their respective descriptions in table 11.1.

Table 11.1 : Various Design Patterns and Their Description

1.	**Cycle**	A pattern that returns the user to a previously visited content node (or Web Page).
2.	**Web Ring**	A pattern that implements a grand cycle that links entire hypertexts in a tour of a subject
3.	**Contour**	A pattern that occurs when cycles impinge upon one another, allowing navigation across paths defined by the cycles.

4.	**Counterpoint**	A pattern that adds hypertext commentary
5.	**Mirrorworld**	Content is presented using different narrative threads, each with a different point of view or perspective.
6.	**Sieve :**	A pattern that guides a user through a series of decisions in order to direct the user to specific content indicated by the sequence of decisions chosen or decisions made.
7.	**Neighborhood :**	A pattern that overlays a uniform navigational frame across all Web pages in order to allow a user to have consistent navigation guidance regardless of location within the WebApp.

(B) Navigation Design

Once a required WebApp architectural structure is chosen and established, next step to be performed in the sequence is the navigation design.

Goal : The main goal of the navigation design is to define navigation pathways which enable a user to access WebApp contents and services.

Description : To achieve this goal, two tasks are required to be followed initially :

1. identify the Semantics of Navigation (SON! for different users of the site, and,
2. define the syntax of achieving the navigation.

Therefore, to accomplish these tasks, firstly the user roles are defined. As such there can be three types of users :

- Visitor may have limited access to the contents.
- Registered customer may have much access to the contents, information, and services.
- Privileged user may have the complete access to the contents.

After finding the various user roles, the WebApp designer creates a Semantic Navigation Unit (SNU) for each goal associated with each user role. So, in this manner, a SNU is created for each goal. According to Gnaho and Lurcher, SNU can be defined as :

"The structure of a WoN is composed of a set of navigational sub-structures that we call Ways of Navigating (WoN). A WoN represents the best navigation way or path for users with certain profiles to achieve their desired goal or sub-goal. Therefore, the concept of WoN is associated to the concept of user profile.

The structure of a WoN is made out of a set of relevant Navigational Nodes (NN) connected by navigational links, including sometimes other SNUs.

That means, that SNUs may themselves be aggregated to form a higher-level SNU, or may be nested to any depth."

In the beginning of navigation design, the WebApp structure is assessed to determine one or more WoNs for each user goal and WoNs are organized into SNUs. As design proceeds, the mechanics syntax of navigation links are identified, like text-based links, icons, buttons etc. Additionally, the designer should also establish appropriate navigation conventions and aids, like, for text-based navigation, color should be used to indicate navigation links and to provide an indication of links already traveled.

(C) Interface Design

The last task of the technical design activity is to perform the interface design which basically concerns with the interfacing between the user and the WebApp.

Goal: The main goal of the interface design is to design the interface as a user-friendly interface, so that a user can easily interact with the WebApp and feels comfort.

Description: To accomplish this goal, we should first know the user interface of a WebApp as it is considered as the "first impression" on the end-user. So, if the user interface is attractive, user-friendly, and, easy to use, learn, and understand then user will obviously like to use that WebApp. Whereas a poorly designed interface can disappoint the user and may cause the user to go elsewhere. Nielsen and Wagner suggested some guidelines based on their on their redesign of a major WebApp. These are listed below.

1. Saver errors, even minor ones, are likely to cause a user to leave die Website and look elsewhere for information or services.
2. Reading speed on a computer monitor is approximately 25 percent slower than reading speed for hard copy. Therefore, do not force the user to read voluminous amounts of text.
3. Avoid "under construction" signs.
4. Usually users prefer not to scroll. Important information should be placed within the dimensions of a typical browser window.
5. Navigation menus and head bars should be designed consistently and should be available on all pages that are available to the user.
6. Aesthetics should never supersede functionality.
7. Navigation options should be obvious, ever to the casual user. The user should not have to search the screen to determine how to link to other contents or services.

Finally design reports are written to introduce and document engineering and scientific designs. In general, these reports have two audiences. One audience

includes other engineers and scientists interested in how the design works and how effective the design is. Another audience includes management interested in the application and effectiveness of the design. Commonly used organization for design reports:

- Summary
- Introduction
- Discussion
- Conclusions
- Appendices.

Summary

The summary, sometimes labeled the abstract or executive summary, is a concise synopsis of the design itself, the motivation for having the design, and the design's effectiveness. The author should assume that the reader has some knowledge of the subject, but has not read the report. For that reason, the summary should provide enough background that it stands on its own. Note that if the summary is called an abstract, you are usually expected to target a technical audience in the summary. Likewise, if an executive summary is requested, you should target a management audience in the summary. For an example summary, see the following "Executive Summary."

Introduction

The "Introduction" of a design report identifies the design problem, the objectives of the design, the assumptions for the design, the design alternatives, and the selection of the design being reported. Also included for transition is a mapping of the entire report. Note that in longer reports, the selection of design is often a separate section. For an example, see the following "Introduction."

Discussion

The discussion presents the design itself, the theory behind the design, the problems encountered (or anticipated) in producing the design, how those problems were (or could be) overcome, and the results of any tests on the design. Note that this part usually consists of two, three, or four main headings. In regards to the actual names of these headings, pay close attention to what your instructor requests. Also consider what would be a logical division for your particular design. For an example section, see the following"discussion."

Conclusions

The "Conclusions" section summarizes the design and testing work completed and assesses how well the design meets the objectives presented in the "Introduction." Note that if the design does not meet the objectives, you should analyze why the design did not succeed and what could be modified to make the

design a success. Besides summarizing the work and analyzing whether the objectives were met, the "Conclusions" section also gives a future perspective for how the design will be used in the future. For an example, see the following "Conclusions."

Appendices

In a design report, appendices often are included. One type of appendix that appears in design reports presents information that is too detailed to be placed into the report's text. For example, if you had a long table giving voltage-current measurements for an RLC circuit, you might place this tabular information in an appendix and include a graph of the data in the report's text. Another type of appendix that often appears in design reports presents tangential information that does not directly concern the design's objectives.

If the appendix is "formal," it should contain a beginning, middle, and ending. For example, if the appendix contains tables of test data, the appendix should not only contain the tabular data, but also formally introduce those tables, discuss why they have been included, and explain the unusual aspects that might confuse the reader. Because of time constraints, your instructor might allow you to include "informal" appendices with calculations and supplemental information. For such "informal" situations, having a clear beginning, middle, and ending is not necessary. However, you should still title the appendix, place a heading on each table, place a caption beneath each figure, and insert comments necessary for reader understanding.

In engineering, technical documentation refers to any type of documentation that describes handling, functionality and architecture of a technical product or a product underdevelopment or use. The intended recipient for product technical documentation is both the (proficient) end user as well as the administrator / service or maintenance technician. In contrast to a mere "cookbook" manual, technical documentation aims at providing enough information for a user to understand inner and outer dependencies of the product at hand.

If technical writers are employed by the technology company, their task is to translate the usually highly formalized or abbreviated technical documentation produced during the development phase into more readable, "user-friendly" prose.

The documentation accompanying a piece of technology is often the only means by which the user can fully understand said technology; regardless, technical documentation is often considered a "necessary evil" by software developers. Consequently, the genre has suffered from what some industry experts lament as a lack of attention and precision.

Questions

Very Short Questions :

1. What do you mean by template ?
2. What is design patterns ?
3. What about the summary in design report ?
4. What is non-technical design ?

Short Questions :

1. What do you mean by Grid Structure ?
2. Give defination of Technical Design.
3. What do you mean by content design ?
4. Describe Networked or Pure Web Structure ?

Long Question :

1. What is the difference between Technical and Non-Technical Design ?
2. What is Non-Technical and Technical Design ? Deploy all parts of Technical System by using Detailed Design.
3. What is Navigation Design?
4. What is Interface Design ?
5. List various design patterns of Technical Design.

❑❑❑

12

Detailed System Design

Once the scope and general configuration of the MIS have been established, the detailed design of the system may be started. A step-by-step explanation of a how-to-do-it procedure for detailed system design applicable to all systems is impractical, for the following reasons :

1. There is a wide variety of approaches to system design in terms of organizing for it, conducting it, and defining its output.
2. A basic question rarely treated by writers deals with the state of the art and to what extent it should be incorporated into systems design. In other words, should we describe management information systems design as it is today, as it could be if the latest state of the art knowledge were utilized, or as it probably will be in five years ?
3. Systems design is a complex of concurrent activities, whereas the nature of our description can proceed along only one line at a time.
4. It is difficult to describe a process of rational design that must be tested in an organization from time to time to determine the likelihood of its being accepted by the members of the organization. That is, the design of a management information system is both a rational and a social process.
5. If it is to represent the design of information systems in general, the explanation may be at such a level or generality that it gives no clues to the multitude of detailed steps involved. On the other hand, the explanation of a detailed procedure can describe only one among thousands and gives no clue to the general nature of design. Attempting to find a middle course sacrifices the advantage of one extreme without gaining much from the other.
6. An explanation of detailed system design procedure must be interrupted frequently by descriptive essays on some aspect with which the designer deals.

Within the limitations on clarity and objectives imposed by these considerations, we will attempt to present the nature of systems design at the "edge of the art." The edge of the art is broad, with part in the present and part in the near future. A general approach will provide the framework, but frequent resort to detailed procedures and descriptions will bring substance to the framework.

Inform and Involve the Organization

The first step in systems design is not a technical one. It is concerned with gaining support for the work that follows. Systems designers must have the support of most members of the organization to obtain information for the design of the system and to obtain acceptance of the final system. At a minimum, members of the organization should be informed of the objectives and nature of the study. It is preferable, if possible, to draw many members into the study, at least in some small way. Furthermore, it is desirable to reassure the employees, if possible, that changes will benefit them or that they will not suffer financially from the implementation of the system. Even so, the natural human resistance to change requires that sufficient information of general progress be disseminated to gradually accustom the employees to their future roles.

The contrary approach – that employees should not be disturbed during the system design – can be quite hazardous. When people are not informed, they seize upon bits of information, construct concepts that may be completely erroneous, and as a consequence often take up detrimental activities. The final system, when announced, may be met with shock, resentment, and both open and covert resistance.

Aim of Detailed Design

The detailed design of an MIS is closely related to the design of operating systems. Sometimes, it is true, the operating system must be accepted without change and a new MIS appended to it. However, it is preferable to design both systems together, and as we discuss the detailed design of the MIS, this parallel effort will be apparent, even though our principal focus is on the MIS.

By drawing upon the analogy of engineering design, we can clarify the meaning of detailed design. The direct goal of engineering design is to furnish the engineering description of a tested and producible product. Engineering design consists of specifications in the form of drawings and specification reports for systems as a whole and for all components in the system. Further, justification documents in the form of reports of mathematical analysis and test results are part of the detailed design. Enough detail must be given so that engineering design documents and manufacturing drawings are sufficient for the shop to construct the product. The production of operating and maintenance instructions is also considered part of the design output.

The analogy of detailed design of MIS readily follows. The aim of the detailed design is to furnish a description of a system that achieves the goals of the

conceptual system design requirements. This description consists of drawings, flowcharts, equipment and personnel specifications, procedures, support tasks, specification of information files, and organization and operating manuals required to run the system. Also part of the design is the documentation of analysis And testing, which justifies the design. The design must be sufficiently detailed that operating management and personnel can implement the system. Whereas conceptual design gives the overall performance specifications for the MIS, the detailed design yields the construction and operating specifications.

Project Management of MIS Detailed Design

Any effort that qualifies as a System design has the dimensions of a project. The first step in the detailed design is therefore a planning and organizing step. For small projects, all phases may be planned for, as described in Chapter 6, before the conceptual (feasibility) design is undertaken. Often, in large projects, not enough is known about the prospective system in advance of the conceptual design to plan for the detailed design project. Further, if the conceptual design indicated that a new system design is not appropriate at this time, any project planning for the detailed design in advance would be wasted.

Once the project manager and key project personnel have been designated, the steps in project management fall into two classes: planning and control. The amount of effort expended in each steals obviously a function of the size of the MIS project and the cost of developing the detailed design of the project. The key steps in planning and control of detailed design, based upon Chapter 6, are recapitulated here.

Project Planning

1. Establish the project objectives. This involves a review, subdivision, and refinement of the performance objectives established by the conceptual design.
2. Define the project tasks. This identifies a hierarchical structure of tasks to be performed in the design of the MIS and may be documented by work package instructions for large projects.
3. Plan the logical development of sequential and concurrent tasks and task activities. This usually requires a network diagram of events and activities.
4. Schedule the work as required by management - established end date and activity network constraints. Essentially, the work and schedule are tied together by completion of the PERT diagram.
5. Estimate labor, equipment, and other costs for the project.
6. Establish a budget for the project by allocating funds to each task and expenditures month by month over the life of the project.
7. Plan the staffing of the project organization over its life.

Project Control

1. Determine whether project objectives are being met as the project progresses.
2. Maintain control over the schedule by changing work loads and emphasis as required by delays in critical activities.
3. Evaluate expenditure of funds in terms of both work accomplished and time. Revise the budget as required to reflect changes in work definition.
4. Evaluate work force utilization and individual work progress, and make adjustments as required.
5. Evaluate time, cost, and work performance in terms of schedules, budgets, and technical plans to identify interaction problems.

Identify Dominant and Trade-off Criteria

Dominant criteria for a system are those that make an activity so important that it overrides all other activities. For example, a dominant criterion might be that the systems operate so that there is never a stockout. This overrides the criterion of minimizing inventory cost. Such a criterion might hold for a company selling human blood, life-preserving drugs, or electric power. It might even hold for a company selling a consumer product where loss of a customer is permanent and all competitors have a no-stockout policy.

Examples of other dominant criteria might be one-day customer service, zero-deject product, specified price range for products, maintenance of multiple sources of supply for all materials and components purchased, or conformity of all research and engineering to long-range corporate plans. It is obvious that identification of the dominant criteria is necessary before subsequent design steps can proceed.

Trade-off criteria are those in which the criterion for performance of an activity may be reduced to increase performance of another activity. For example, the criterion of low manufacturing costs might be balanced against that of long-range public image of the firm achieved by reduction in environmental pollution. Again, the criterion of producing styles or models for many segments of the market might be balanced against that of maintaining low manufacturing and service costs.

The reason for identifying dominant and trade-off criteria is that as the detailed design is developed, decision centers (managers or computers) must be identified to achieve such criteria or make trade-offs. The MIS must be designed to provide the information for the decisions, or at lower and programmed levels, to make the trade-offs.

Define The Subsystems

We start the process of defining the subsystems with two principal blocks of Information: (1) the conceptual design and (2) the dominant and trade-off

performance criteria. Although the conceptual design requires some assumptions concerning the subsystems, it is necessary now to review these subsystems and to redefine them if it seems appropriate. Based upon the conceptual design, investigation of the detailed activities of each major activity block must be undertaken. Consider, for example, the conceptual design representation of a business system given in Figure 12.1. Each large block (or system) must be broken down to determine all activities required and the necessary information inputs and outputs of each activity. Careful analyses of such activities is critical in detailed design. Figure 12.3 shows a typical form, which helps to develop descriptions of each activity.

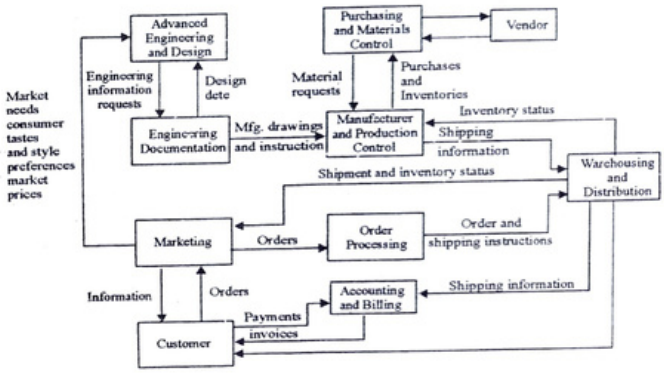

Figure 12.1 Conceptual System design for Business System

The quality control activity, one of many in the manufacturing subsystem, might be analyzed as shown in Figure 12.3. The activity processor is made up of equipment and personnel assigned. One form of information output is the revised figure for resources available after commitment to the activity. The operating input is a batch or "lot" of parts, and the output consists of reduced-size batches of a statistically specified quality plus defectives that have been separated out. The information results give the average quality of batches, the number of rejects, and the number of batches inspected per unit of time. The kind of information output captured must be based upon decisions to be made for planning and control. At this stage, only speculations on needed information for an activity may be made.

ACTIVITY (TRANSACTION)		NETWORK DIGRAM ACTIVITY NUBER
PURPOSE AND DESCRPTION		
INPUTS OUTPUT	MEDIA MEDIA	
SEQUENCE OF ELEMENTS OF ACTIVITY	PERFORMANCE	DECISION RULE

Figure 12.2 Activity Design Form

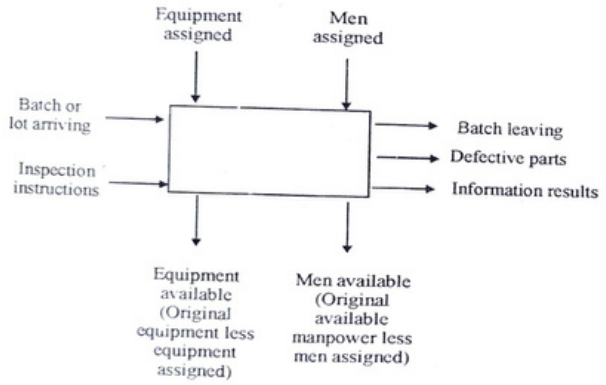

Figure 12.3 Model of Activity Operation and Information

The information system must be based upon the operating system. Once this operating system is outlined by the selection of the general concept, certain basic relationships among major activities become more or less fixed. However, there is still considerable freedom in establishing the detailed activities and their relationships. The detailed activities, once defined, may be related in network form, as shown in Figure 12.4.

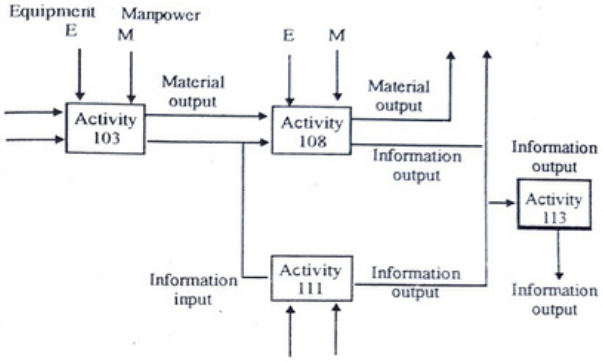

Figure 12.4 Interrelating Activities

The degree of breakdown of the major activities, of course, determines the size and complexity of the network. If the activities are broken down too finely, the design will never be completed. If a major activity is broken down too coarsely, vital material, information, and decision needs will not be factored into the design. Furthermore, optional rearrangement or regrouping of activities will not be examined. If we could conceive of a hierarchy of activities such as

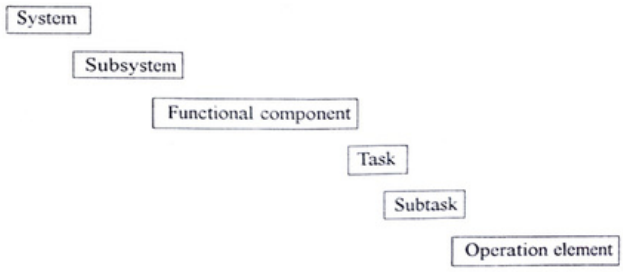

Fig 12.5 Hierarchy of activities

Then we would probably want to develop our activity network at the "functional component" level. Once activity networks have been developed to include each major activity of the conceptual design, the subsystems are then redefined. A subsystem may consist simply of the activities corresponding to a major block activity of the conceptual design, or some detailed activity blocks may be transferred from one group to another to make up the network of the subsystem. Any changes such as this will require a redefining of the major activity block in terms of its performance requirements. Quite often, however, a major block activity must be considered as comprising several subsystems. Grouping of activities into a subsystem may be based upon various consideration, such as (1) common functions, (2) common techniques or procedures, (3) logical flow relationships, or (4) common inputs or outputs.

The revised subsystems can be identified on the network diagrams by drawing a loop around the aggregate of activities to be included in a particular subsystem. At this point, all connecting lines for activities that cross over the loop represent either inputs or outputs to the subsystems. The loop itself is a boundary of the subsystem and an interface between two or more subsystems.

Because organizational authority and responsibility structures are often divided by subsystem boundaries, problems of interfacing subsystems require careful attention, that is, inputs and outputs between interfacing systems must be matched.

The approach that we have given here for identifying subsystems is analo-gous to one method of development of a new company's organization structure by analysis followed by synthesis. This consists of identifying activities and tasks, grouping tasks into positions, and then grouping positions into components on some rational basis. An alternative approach that is quicker but not so thorough is simply to divide the major activities of the conceptual design into the subsystems apparently required to fulfill the major activities. Such a procedure may lead to incompletely defined subsystems, mismatches between subsystems, and missing activities.

Information for Defining Subsystems

The objective of the design search is to find a set of subsystems that satisfies the performance requirements specified by the conceptual design. To do this, we must search for information that helps us select and define the subsystems. Such information consists of

1. Dominant and trade-off criteria for the operating and the MIS systems as a whole. Dominant criteria—such as the decision to utilize only current manufacturing area available, to use company salespersons rather than manufacturer's representatives, to utilize current computing facilities, or to install time-sharing terminals for executive use—impose some clear limitations on alternatives for the designer. Where dominant criteria are absent, trade-offs that permit different emphasis on different functional

activities must be identified, so that subsystems can be defined realistically. Analysis may indicate, for instance, that the logistics activities should be separated from both manufacturing and marketing management for greatest effectiveness and efficiency. This will require the study and development of a logistics system closely coupled to the other two systems.

2. Available resources the company will commit. Systems must be designed, obviously, in terms of what is available to implement them.
3. Required activities for achievement of systems operations and performance specifi-cations. Each activity and its relationship to other activities must be identified.
4. Necessary control positions in the system. Every system has a hierarchical structure of control points, usually corresponding to the organizational responsibility struc-ture. However, organizational responsibility for control of variances is often not well defined. The MIS designer must get control positions clarified to develop informa-tion flows.
5. Management decision points for system planning and control. At the upper levels of the organization, important decisions must be made regarding system operation and variances from major company goals. These top-management positions and their information requirements must be identified.
6. Information required for programmed decision making. Complete information requirements for decisions capable of being processed by decision tables and models with the aid of the computer must be uncovered.
7. Specific output requirements of all systems. This includes a detailed list of purposes to be served by the information output. The specific content of each report or communication and the method of utilization should be determined. The frequency of reports, their formats (written, visual, audio, etc.), distribution, and filing must be established. This information assists with the development of activity descriptions just discussed. Such information may be recorded on the activity form shown in Figure 12.2.

Obtaining Information

The designer utilizes four principal sources for the design of the MIS. These are

1. Task force meetings
2. Personal interviews
3. Internal and external source documents
4. Personal observation of operations and communications, when feasible

Task Force Meetings

For the design of large systems, the use of task forces for the development of information and ideas is usually advantageous. The task force for a single major

activity block should consist of both managers and key specialists. The designer should chair the task force meetings. The designer's function is to draw out ideas and information, synthesize ideas, including his or her own, and present in diagrams and documents the synthesis for evaluation and modification. The task force meetings serve to bring out information gaps, operating needs, and controversial points. In repeated meetings, the design of a subsystem is hammered out.

Interviews

Instead of, or supplemental to, task force meetings, the designer should conduct interviews with key managers at top and intermediate levels, with key specialists, and with a sampling of operating employees. Although it appears obvious, it cannot be emphasized enough that the designer must use tact in interviews. Whether interviewing the top manager or the lowest-level employee, the designer's role is that of a searcher for knowledge, not a lecturer on systems.

In interviews with managers, the designer should seek information on

1. Objectives of the firm or organizational component
2. Major policies in force or needed to accomplish these objectives
3. The categories of information the managers desire
4. Speed of access to the various categories of information desired by managers
5. Intervals of time desired between receipt of various types of information
6. Format desired for information presented
7. Style of decision making of the managers
8. Resources that will be committed for the implementation and operation of the system
9. Degree of manager involvement in classes of decisions: individual decision making, participative decision making, or partially routine (programmed) decision making
10. Organizational relationships that would facilitate system operation and manage-ment decision making

Systems designers should not expect too much from managers in the way of defining information needs. Rather, they should work with managers to identify objectives and develop plans. The identification of necessary information will follow from this.

Internal and External Source Documents

The use of internal source documents primarily provides the systems designer with a point of departure. From a practical viewpoint, it is likely that many traditional operations and reports must be retained simply to provide some continuity of operations; and this is appropriate, because the current methods are usually the result of continued minor improvements. Modification or regrouping of activities or

data batches is still feasible for the systems designer, however. The number of internal source documents may be great, depending upon the company, so that no complete listing is given here. Organization and policy guides, procedures manuals, master budgets and account structure, and the many functional reports of engineering, manufacturing, marketing, purchasing, and employee and public relations should be examined according to their relevancy to the system being designed. Sometimes, reports such as records of customers' complaints or of service calls may provide the key to system design needs.

External source documents provide economic, marketing, industry, and financial information related to the firm that may be of assistance. A review of the company by a securities analyst in a financial journal may provide very valuable insights.

Direct Observation

Designers should not isolate themselves in their offices to sketch out designs; they should make on-the-spot surveys of operations in action. It would be foolish, for example, to develop a systems design in which supervisors in the factory provide hourly personal reports to a planning/control center if the factory were a half-mile in length or if the planning/control group were in a building some distance away. Similar absurd situations could be conjectured for sales reports or physical distribution reports. On-site inspection will reveal the physical and environmental restraints that may have to be accepted in a new system. Conversely, such inspection may also lead to a major revision of the physical facilities to suit the system design.

Designers should record, as they go along, the relevant (and probably much irrelevant) data they are gathering. In the case of large documents, they would, of course, merely make notes of points vital to their investigation. At the end of certain phases, they should try to organize the data so that future information may be related to that already gathered. Finally, at some point, they sketch alternative designs for the operating subsystems.

Sketch The Detailed Operating Subsystems And Information Flows

The development of the detailed design is first carried out for the subsystem, functional, and task levels of detail. It is very similar to detailed engineering design, which requires trial and error, shifting operations to find good arrangements, and performing calculations to check out the system. The equivalents of engineering sketches in MIS design are the flowcharts. Chapter 3 shows the graphic symbols and their meanings as used by designers. There are three types of systems flowcharts :

1. Task-oriented charts. These are block diagrams showing the relationships among the various tasks or activities. (See Figure 8.5) Subsequently the detailed elemental steps required to complete an activity are analyzed and described step by step on an operation analysis form (sometimes called a flow-process chart).

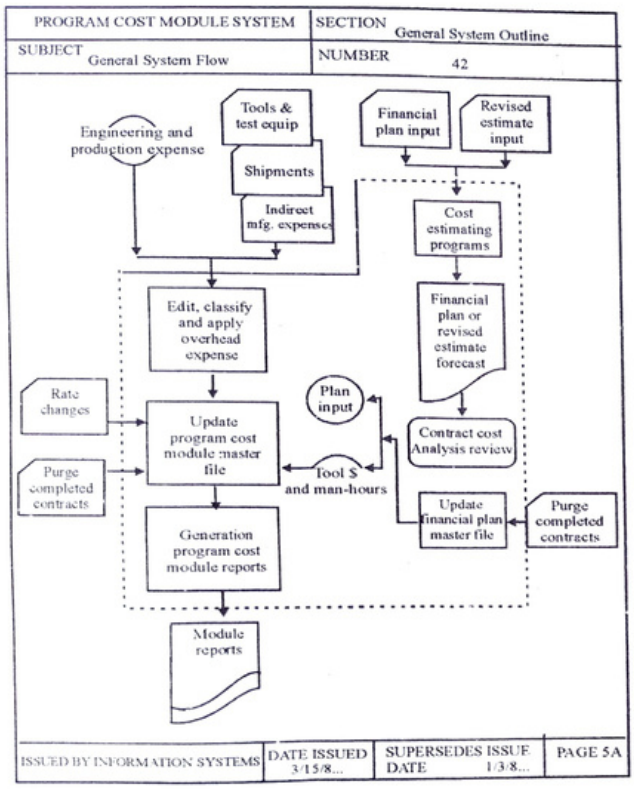

Fig. 12.6 Flowchart: Task Oriented

2. Forms-oriented charts. These charts identify the forms used in communicating or reporting and trace the flow of all copies through the organization. In some cases, the chronological movement may receive emphasis.

3. Program flowcharts (block diagrams). Prepared by the people who give instructions to the computer, the program flowchart is a fundamental tool of programming, designed to show the logical sequence of steps to be carried out by the computer. It structures the logic that the cooling of the programs will follow. Program flowcharts will be of interest at a later stage in the design, but we include them here for completeness.

The flowcharts are not the complete detailed design. They show primarily flows and relationships. Inputs and outputs are shown only in gross form. The quantitative relations among elements in the systems must be expressed in terms of mathematical models. Where this is not possible, detailed verbal descriptions must be used to actually develop the detailed operating design. The flowcharts are important, however, in developing the information necessary for managerial decisions with respect to the design for model constructions, and for programmed decision making in system operation.

Determine The Degree of Automation of Each Operation

Each operation in the flowchart should next be examined to establish the level of automation possible and the level desired. We can automate most processes to different degrees, depending on management desires :

1. No automation. People continue to do the work with at most manual aids. (A mechanic uses a wrench to put a nut on a bolt.)
2. Work automated, control manual. Electromechanical devices now do the work, but they must be caused to do the work by some human intervention. (A tool and die maker sets up a lathe to shave off a thousandth of an inch, then starts the process, and leaves.)
3. Work automated, feedback automated, control manual. We have added a feedback loop that gives the operator information but stiff relies on manual intervention to modify the process. (A badge reader that controls door access opens the door for valid badges and sounds an alarm in the security office for invalid badges.)
4. Work automated, feedback automated, control automated. This system requires little or no human intervention after it is installed and started. (Many power management systems—air conditioning, heating, lights, humidity control—are fully automated and "correct*" themselves as they operate.)

It is important to remember here that the highest possible degree of automation may not be the best for the firm and task in question. Each aspect of the MIS should be in support of the company's objectives and should mesh with manage-ment's desires for the function being automated.

Widely contrasting levels of automation in a system may be suspect and should be examined. It is not at all incorrect, however, to choose an "all manual" (low-level automation) system. Low-level automation may be preferred when

1. Problems are not well structured
2. The decision criteria cannot be well defined
3. The rules for making decisions must be constantly modified or changed
4. There are gaps in the data entering the system
5. The data entering the system are ambiguous, inconsistent, or somewhat unreliable
6. The processing steps are simple and few in number
7. The cost of human labor is low relative to the cost of the equipment
8. Storage of data is negligible

Inform and Involve The Organization Again

Although it is important not to disrupt the company's main business as we build the MIS, this is an ideal time to do some work that increases the acceptance of the MIS being designed. Upper management should be given a brief overview of the design and a status of the design effort as a project. Lower-level manage-ment should be shown the system and subsystem flowcharts, requested to comment on their accuracy, and solicited for guidance on the degree of automa-tion desired in each case. Although this may generate lots of argument and conflicting comments and guidance, it is a valuable tool for determining which managers are supportive and which ones are resisting change; it also starts to generate feelings of acceptance and ownership by the mere facts of participation and involvement. And, finally, good ideas and suggestions almost always emerge. Last, but by no means least important, explain the system in detail to the final users of the system/subsystem. Although this may be a tedious process, acceptance at this level is critical. If this is approached from an "I need your help and guidance" point of view, a flood of detailed suggestions will come out, pitfalls to avoid will be pointed out, and the acceptance and involvement process will have been started.

The activities just discussed will help to avoid the fate of many manage-ment information systems : "It's their system, they designed it; let them make it work!"

Inputs, Outputs, and Processing

Armed with the flow diagrams for each subsystem and a substantial amount of constructive criticism from the future users of the MIS, we are now prepared to put more flesh on the design. We turn to defining the subsystem inputs and outputs in more detail and to providing a better description of how each subsystem will perform its task.

Inputs and Outputs: Forms

Each subsystem has requirements for information coming in and produces outgoing information to be used by another subsystem or an end user. These inputs and outputs are often forms, especially when forms are considered in the light of

their modern definition: any standardized communication that is an essential link in an operating procedure is the equivalent of a form. This covers paper forms such as the IRS 1040 tax form, punched cards such as those used by many large companies for billing, envelopes, mailing labels, microfilm, and computer printouts of sales analysis reports.

Two tasks face us here :

1. Specify the inputs and outputs exactly. Also show where the information can be obtained or to whom it will go.
2. Design forms that fulfill their function and are easy to use.

The first of these tasks is fairly straightforward but not so easy to accomplish. Existing systems must be studied in detail, and many people must be interviewed to do a thorough job. Anything less than a thorough analysis of inputs and outputs exposes the system to not being able to get the input or not being able to provide the required output.

The second task, designing useful forms, is difficult. We must first identify

- The function of the form
- When the form is used
- How many copies are used in a single cycle for the transaction, transmission, and storage
- Who fills out the forms and who uses the form
- How many units of the form are required per year

A number of specific factors should be considered in forms design :

1. After all the following factors have been considered, the form should be attractive and easy to read.
2. Most forms should have a title. Examples of those that do not may be checks, labels, tags, and video displays. The title should be specific enough so that the function of the form may be determined, in most cases, without seeing the form.
3. Forms should have an identification number with the date of issuance.
4. Group information into related areas on the form. Use a box design with captions printed in small, distinct type in the upper left-hand corner of the boxes.
5. Arrange the items in that there is a logical flow from left to right and from top to bottom in filling out the form. Reporting forms usually have summaries in the right columns and bottom rows. Shading may also be used to make report forms more readable.
6. The use of box items to be checked or coded improves the readability of the design. When extreme emphasis is on accuracy, large boxes, heavily blocked sections, and lots of open space help.

7. Provide sufficient space for entering data and do not bleed lines off the sides of the forms. If horizontal lines stop short of the edge of the sheets, it is less likely that variable data will run off the edge.
8. Consider colored ink for captions to make variable data stand out. Colored ink may also be used for serial identification numbers.
9. "Boiler plate" or standard contract information is often on the back of forms such as purchase orders. Alternatively, detailed instructions on how to fill out the form may be printed on the back.
10. Routing instructions for each copy may be indicated by using a different color paper for each copy and printing routing instructions on the margin.
11. Type faces, the use of heavy and light lines, shaded areas, and color should be combined to provide an aesthetic form that makes the variable data stand out.
12. Standard form sizes should be used. These range from label sizes and shapes to the usual 8½" x 11" forms to 11" x 17" foldouts. If forms are kept small, they may be cramped in appearance, difficult to handle, and awkward to file. On the other hand, cost savings may be realized; since less paper or cardboard is required. Size is also related to the standard equipment available to process the forms—for example, sorters for checks at banks.
13. If the form is to be placed in a binder, leave adequate blank space at the top or sides.
14. If forms and envelopes are designed to match, the location of the address shown on the form must show through the envelope window with normal folding.
15. When multiple copies of a form are desired, they may be obtained in several ways. Multiple sheets may have carbon interleaves, or special no-carbon-required paper may be used. The form may be a ditto master from which, after the variable data have been added, copies may be run off. (This is appropriate for five or more copies.) Finally, a single copy of the form may be completed and then copies run off on an office copier.
16. Formats to be shown on video terminals should separate blocks of information adequately. If too many data are attempted in one display, they will be difficult to read. It is no problem to show a series of formats for clear presentation.

Even with all the care taken in designing the forms to be used in input and output, we must be sure to test the forms thoroughly by discussing them with the users and letting them actually try using the forms.

Inputs and Outputs: Database

The database is the data that must be obtained (sometimes called "captured") and usually stored for later retrieval for managerial decision making. It

also consists of data that will be utilized in programmed decision making and real-time control. As discussed in Chapter 4, the database is derived from the needs of management for information to guide the total business system. We start with the study of management's problems and information needs, develop flowcharts of systems that meet management's requirements, and then detail the data requirements of the systems.

A systematic approach to the development of the database is as follows :

1. Identify all points on the flowcharts that require data inputs. Generally these are inputs to activities or transactions as well as decision tables or modeling operations
2. Prepare a data or file worksheet for each data element, giving
   a. Source of data
   b. Length and format
   c. Current and potential frequency of updating
   d. Retention schedule for the data e. End use of the data
3. Group all data worksheets by system and check for omissions.
4. Group all data worksheets by activity and by organizational component and note duplications.
5. Eliminate duplicated data requirements to develop an integrated database for which cross-functional use of the master file is employed.
6. Evaluate the items in the master file for frequency of need and value of the data to the system versus cost of obtaining the data. Judgment must now be used to prune the file for possible revision of the system design if the cost of file construction and file maintenance exceeds the estimated value of the system or the available resources.

The database deserves careful attention because of its cost and operational impacts on the MIS and the larger organization. Chapter 4 discussed database management in some detail. The reader may wish to review that chapter.

Processing

A great deal of information has been gathered since the flowcharts for the subsystems were sketched. Inputs and outputs have been detailed. Numerous users have reviewed those flowcharts. At this point, corrections should be made to the subsystem flowcharts. In addition, each task specified inside the subsystem should be broken down further in either mathematical or narrative format. This process ties all the pieces of information we have gathered back to the original MIS goal. It also serves to reverify the detailed design and add flesh to the outline.

1. Problems are not well structured
2. The decision criteria cannot be well defined
3. The rules for making decisions must be constantly modified or changed
4. There are gaps in the data entering the system
5. The data entering the system are ambiguous, inconsistent, or somewhat unreliable
6. The processing steps are simple and few in number
7. The cost of human labor is low relative to the cost of the equipment
8. Storage of data is negligible

Inform and Involve The Organization Again

Although it is important not to disrupt the company's main business as we build the MIS, this is an ideal time to do some work that increases the acceptance of the MIS being designed. Upper management should be given a brief overview of the design and a status of the design effort as a project. Lower-level manage-ment should be shown the system and subsystem flowcharts, requested to comment on their accuracy, and solicited for guidance on the degree of automa-tion desired in each case. Although this may generate lots of argument and conflicting comments and guidance, it is a valuable tool for determining which managers are supportive and which ones are resisting change; it also starts to generate feelings of acceptance and ownership by the mere facts of participation and involvement. And, finally, good ideas and suggestions almost always emerge. Last, but by no means least important, explain the system in detail to the final users of the system/subsystem. Although this may be a tedious process, acceptance at this level is critical. If this is approached from an "I need your help and guidance" point of view, a flood of detailed suggestions will come out, pitfalls to avoid will be pointed out, and the acceptance and involvement process will have been started.

The activities just discussed will help to avoid the fate of many manage-ment information systems "It's their system, they designed it; let them make it work!"

Inputs, Outputs, and Processing

Armed with the flow diagrams for each subsystem and a substantial amount of constructive criticism from the future users of the MIS, we are now prepared to put more flesh on the design. We turn to defining the subsystem inputs and outputs in more detail and to providing a better description of how each subsystem will perform its task.

Inputs and Outputs: Forms

Each subsystem has requirements for information coming in and produces outgoing information to be used by another subsystem or an end user. These inputs and outputs are often forms, especially when forms are considered in the light of

their modern definition: any standardized communication that is an essential link in an operating procedure is the equivalent of a form. This covers paper forms such as the IRS 1040 tax form, punched cards such as those used by many large companies for billing, envelopes, mailing labels, microfilm, and computer printouts of sales analysis reports.

Two tasks face us here :

1. Specify the inputs and outputs exactly. Also show where the information can be obtained or to whom it will go.
2. Design forms that fulfill their function and are easy to use.

The first of these tasks is fairly straightforward but not so easy to accomplish. Existing systems must be studied in detail, and many people must be interviewed to do a thorough job. Anything less than a thorough analysis of inputs and outputs exposes the system to not being able to get the input or not being able to provide the required output.

The second task, designing useful forms, is difficult. We must first identify

- The function of the form
- When the form is used
- How many copies are used in a single cycle for the transaction, transmission, and storage
- Who fills out the forms and who uses the form
- How many units of the form are required per year

A number of specific factors should be considered in forms design :

1. After all the following factors have been considered, the form should be attractive and easy to read.
2. Most forms should have a title. Examples of those that do not may be checks, labels, tags, and video displays. The title should be specific enough so that the function of the form may be determined, in most cases, without seeing the form.
3. Forms should have an identification number with the date of issuance.
4. Group information into related areas on the form. Use a box design with captions printed in small, distinct type in the upper left-hand corner of the boxes.
5. Arrange the items in that there is a logical flow from left to right and from top to bottom in filling out the form. Reporting forms usually have summaries in the right columns and bottom rows. Shading may also be used to make report forms more readable.
6. The use of box items to be checked or coded improves the readability of the design. When extreme emphasis is on accuracy, large boxes, heavily blocked sections, and lots of open space help.

7. Provide sufficient space for entering data and do not bleed lines off the sides of the forms. If horizontal lines stop short of the edge of the sheets, it is less likely that variable data will run off the edge.
8. Consider colored ink for captions to make variable data stand out. Colored ink may also be used for serial identification numbers.
9. "Boiler plate" or standard contract information is often on the back of forms such as purchase orders. Alternatively, detailed instructions on how to fill out the form may be printed on the back.
10. Routing instructions for each copy may be indicated by using a different color paper for each copy and printing routing instructions on the margin.
11. Type faces, the use of heavy and light lines, shaded areas, and color should be combined to provide an aesthetic form that makes the variable data stand out.
12. Standard form sizes should be used. These range from label sizes and shapes to the usual 8½" x 11" forms to 11" x 17" foldouts. If forms are kept small, they may be cramped in appearance, difficult to handle, and awkward to file. On the other hand, cost savings may be realized; since less paper or cardboard is required. Size is also related to the standard equipment available to process the forms—for example, sorters for checks at banks.
13. If the form is to be placed in a binder, leave adequate blank space at the top or sides.
14. If forms and envelopes are designed to match, the location of the address shown on the form must show through the envelope window with normal folding.
15. When multiple copies of a form are desired, they may be obtained in several ways. Multiple sheets may have carbon interleaves, or special no-carbon-required paper may be used. The form may be a ditto master from which, after the variable data have been added, copies may be run off. (This is appropriate for five or more copies.) Finally, a single copy of the form may be completed and then copies run off on an office copier.
16. Formats to be shown on video terminals should separate blocks of information adequately. If too many data are attempted in one display, they will be difficult to read. It is no problem to show a series of formats for clear presentation.

Even with all the care taken in designing the forms to be used in input and output, we must be sure to test the forms thoroughly by discussing them with the users and letting them actually try using the forms.

Inputs and Outputs: Database

The database is the data that must be obtained (sometimes called "captured") and usually stored for later retrieval for managerial decision making. It

also consists of data that will be utilized in programmed decision making and real-time control. As discussed in Chapter 4, the database is derived from the needs of management for information to guide the total business system. We start with the study of management's problems and information needs, develop flowcharts of systems that meet management's requirements, and then detail the data requirements of the systems.

A systematic approach to the development of the database is as follows :

1. Identify all points on the flowcharts that require data inputs. Generally these are inputs to activities or transactions as well as decision tables or modeling operations
2. Prepare a data or file worksheet for each data element, giving
   a. Source of data
   b Length and format
   c. Current and potential frequency of updating
   d. Retention schedule for the data e. End use of the data
3. Group all data worksheets by system and check for omissions.
4. Group all data worksheets by activity and by organizational component and note duplications.
5. Eliminate duplicated data requirements to develop an integrated database for which cross-functional use of the master file is employed.
6. Evaluate the items in the master file for frequency of need and value of the data to the system versus cost of obtaining the data. Judgment must now be used to prune the file for possible revision of the system design if the cost of file construction and file maintenance exceeds the estimated value of the system or the available resources.

The database deserves careful attention because of its cost and operational impacts on the MIS and the larger organization. Chapter 4 discussed database management in some detail. The reader may wish to review that chapter.

Processing

A great deal of information has been gathered since the flowcharts for the subsystems were sketched. Inputs and outputs have been detailed. Numerous users have reviewed those flowcharts. At this point, corrections should be made to the subsystem flowcharts. In addition, each task specified inside the subsystem should be broken down further in either mathematical or narrative format. This process ties all the pieces of information we have gathered back to the original MIS goal. It also serves to reverify the detailed design and add flesh to the outline.

Early System Testing

There are three ways to get early feedback on the viability of the MIS just designed:

- Modeling
- Simulation
- Test planning

We will consider each in turn. Their importance lies in the impact that early problem discovery has on project cost. A problem discovered at design time is much cheaper to fix than is one found after the system has been put into operation.

Modeling the System Quantitatively

During our design efforts, we have quantified as much of the system as possible. We now attempt to determine quantitative ranges for inputs and outputs, quantitative relationships for the transfer functions, and time and reliability responses for operations in the system. Decision models are developed in both mathematical equation form and decision table form. The purpose of modeling at this stage is to define the system more precisely and to improve it.

Besides the mathematical modeling of systems discussed in Chapter 12, logic tables may be developed for decision models. Such "decision tables" may include both quantitative and qualitative bases for decision making. Decision tables are valuable for both design and documentation of systems. Decision structure tables are in the form of

if these conditions exist...
then perform these actions....

The "if listings form the condition stub. The "then" listings make up the action stub. Figure 8-6 shows a simplified decision table for purposes of explana-tion. The first column, under "Decision Rules," is read as follows:

if the car is driven less than 10,000 miles a year and
if the age of the youngest driver is over age 25 and
if no drivers have major physical defects and
if no driver has had more than one accident in the past 3 years and
if no driver has had a speeding conviction and
if the major use for the car is pleasure driving',
then the policy limit is $100,000 / $300,000
and the policy rate is $1.12 / $1000 and our policy identification type is A

Auto Policy Coverage	Decision Rules			
Miles driven per year	10,000	10,000	15,000	15,000
Age of youngest driver	> 25	> 25	25	25
Physical defects f driver (one eye, one arm)	None	None	None	None
Accounts in Last 3 years	1	1	2	2
Speeding convictions	None	None	None	None
Major use	plessure	business	plessure	business
Policy limit, $1,000's	100/300	100/200	50/100	25/50
Policy rate per $1,000	1.12	1.50	1.74	2.50
Type of Policy	A	D	F	H

Fig. 12.7 : Decision Table for Insurance Decisions

Testing the System by Simulation

For very small systems, the best test may be conversion to on-line operations. In very large systems, simulation of the entire system may be too complex and costly. However, for many systems and for subsystems and functional compo-nents of large systems, testing by simulation should be carried out. The alterna-tive, conversion to the new system and debugging and redesigning during change-over, should be held to a minimum. It is costly and damaging to the morale of the people responsible for implementing the system. Simulation has a further ad-vantage in permitting evaluation of the system against the criteria of the conceptual design performance specifications.

The procedure for a simulation test of the whole system is as follows :

1. By random methods, select values of exogenous data from within the anticipated ranges of each variable. Let us illustrate with the input to our business of the variable we identify as share of market held by our principal competitor. The range for this is 20 to 35 percent. By the Monte Carlo technique, described in Chapter 12, we choose a random number and find the corresponding value of our competitor's market share is 23 percent. We do the same for other exogenous inputs, such as economic indexes, interest rates, cost of raw materials, size of new markets, and labor costs. Nonquantitative inputs may be handled just as easily by listing alternatives and selecting one by random number means.
2. Trace the effect of the exogenous inputs through the system. Where the activity processors are automatic, as in the case for computerized systems or decision tables, the process is straightforward. Where humans are

involved, errors and time lags of a random nature may be introduced, or as a first approximation, the humans may be considered to perform routine operations with machinelike efficiency.

There will be numerous points in the system at which human decisions, and more important, managerial decisions, will be required. If the system is to be properly tested, it is necessary to have an appropriate manager make the decision based on the information available from the system at that stage of the simulation. Inadequate information, lack of understanding of the situation by the manager, and inability of the system to control errors in judgment may be uncovered by this realistic type of simulation.

3. Examine outputs of various subsystems. Have cost variables been kept under control? Have outputs been restricted to specified ranges? Will the subsystems be able to respond rapidly enough to inputs so that the entire system will respond in time to maintain itself in its environment? Are all operations being performed according to specifications, or are crisis decisions being made constantly to keep them in line?
4. Repeat steps 1 through 3 several times. It is not feasible to test the whole system in the way that some operations are checked out. That is, a Monte Carlo simulation of a stochastic inventory system may require several thousand simulation cycles, but the process is mechanistic enough so that they can be carried out on a computer.

Planning to Test the MIS

When an individual or set of people lay out detailed plans for verifying the final version of the MIS, errors and omissions often show up. A detailed plan for testing the MIS requires that the following types of questions be asked (and answered in detail):

- Exactly which subsystems are available on various dates?
- What are all of the external (user) interfaces?
- What are some typical scenarios that can be tried out against the MIS?
- Who are some likely test subjects (users and managers)?
- And so on...

The process of asking and answering these questions makes the design more thorough and accurate.

Software, Hardware, and Tools

The software design should be done during this phase so that it will be available to merge with the larger system at the appropriate time. In fact, the coordination of the systems design group and the computer organization should start at the time of the conceptual design. Trained programmers should be on hand

at the start of the detailed design work and at least nine months prior to installation. There are some principal steps in software development for systems over which management, through the systems designers, should maintain surveillance. These steps, carried out by the computer organization, are

1. Develop standards and procedures for programming. Standardized charting sym-bols, techniques, and records should be maintained.
2. Study the conceptual design specifications and work with the system designers in the development of the detailed design. The computer programmers should be a part of the design team by contributing their expertise as needed.
3. Develop the data processing logic and prepare the programming flowcharts. When the programming charts are completed, they should be reviewed by the systems design group.

Similarly, by now the system designers should be considering possible hardware configurations seriously. Without going into detail on how to choose computer hardware, the following items should guide the designers in their deliberations:

1. Buy enough computing power to do the whole job. Buy a little extra computing power, but remember that the MIS department is a support function.
2. Buy enough external storage to hold the required database and any application code planned. Leave some room for growth here also.
3. Buy other peripheral devices only as needed to support the problems the MIS is solving (printers to print the reports, displays to allow data entry, etc.).
4. Buy computer supplies as needed (paper, ribbons, tapes, etc.).
5. Buy enough computer power to provide the required user response.
6. Buy the best documentation.
7. Buy a sufficient maintenance/support package.

Finally, the system designers need to look ahead to the implementation phase and decide if any tools, methodologies, or procedures are needed. For example, if today's records are kept on punched cards and the new MIS has a disk database, a software tool is needed to transfer those records. The tools should be planned for so they will be available when needed.

Propose an Organization to Operate the System

The development of a new organization structure when the company executives and organization are in place is fraught with practical obstacles. Managers would consider it an imposition if the MIS group were to suggest regroupings

involved, errors and time lags of a random nature may be introduced, or as a first approximation, the humans may be considered to perform routine operations with machinelike efficiency.

There will be numerous points in the system at which human decisions, and more important, managerial decisions, will be required. If the system is to be properly tested, it is necessary to have an appropriate manager make the decision based on the information available from the system at that stage of the simulation. Inadequate information, lack of understanding of the situation by the manager, and inability of the system to control errors in judgment may be uncovered by this realistic type of simulation.

3. Examine outputs of various subsystems. Have cost variables been kept under control? Have outputs been restricted to specified ranges? Will the subsystems be able to respond rapidly enough to inputs so that the entire system will respond in time to maintain itself in its environment? Are all operations being performed according to specifications, or are crisis decisions being made constantly to keep them in line?
4. Repeat steps 1 through 3 several times. It is not feasible to test the whole system in the way that some operations are checked out. That is, a Monte Carlo simulation of a stochastic inventory system may require several thousand simulation cycles, but the process is mechanistic enough so that they can be carried out on a computer.

Planning to Test the MIS

When an individual or set of people lay out detailed plans for verifying the final version of the MIS, errors and omissions often show up. A detailed plan for testing the MIS requires that the following types of questions be asked (and answered in detail):

- Exactly which subsystems are available on various dates?
- What are all of the external (user) interfaces?
- What are some typical scenarios that can be tried out against the MIS?
- Who are some likely test subjects (users and managers)?
- And so on...

The process of asking and answering these questions makes the design more thorough and accurate.

Software, Hardware, and Tools

The software design should be done during this phase so that it will be available to merge with the larger system at the appropriate time. In fact, the coordination of the systems design group and the computer organization should start at the time of the conceptual design. Trained programmers should be on hand

at the start of the detailed design work and at least nine months prior to installation. There are some principal steps in software development for systems over which management, through the systems designers, should maintain surveillance. These steps, carried out by the computer organization, are

1. Develop standards and procedures for programming. Standardized charting sym-bols, techniques, and records should be maintained.
2. Study the conceptual design specifications and work with the system designers in the development of the detailed design. The computer programmers should be a part of the design team by contributing their expertise as needed.
3. Develop the data processing logic and prepare the programming flowcharts. When the programming charts are completed, they should be reviewed by the systems design group.

Similarly, by now the system designers should be considering possible hardware configurations seriously. Without going into detail on how to choose computer hardware, the following items should guide the designers in their deliberations:

1. Buy enough computing power to do the whole job. Buy a little extra computing power, but remember that the MIS department is a support function.
2. Buy enough external storage to hold the required database and any application code planned. Leave some room for growth here also.
3. Buy other peripheral devices only as needed to support the problems the MIS is solving (printers to print the reports, displays to allow data entry, etc.).
4. Buy computer supplies as needed (paper, ribbons, tapes, etc.).
5. Buy enough computer power to provide the required user response.
6. Buy the best documentation.
7. Buy a sufficient maintenance/support package.

Finally, the system designers need to look ahead to the implementation phase and decide if any tools, methodologies, or procedures are needed. For example, if today's records are kept on punched cards and the new MIS has a disk database, a software tool is needed to transfer those records. The tools should be planned for so they will be available when needed.

Propose an Organization to Operate the System

The development of a new organization structure when the company executives and organization are in place is fraught with practical obstacles. Managers would consider it an imposition if the MIS group were to suggest regroupings.

particu-larly if an individual manager's position may be abolished. The MIS group should work with incumbent and top managers to suggest organizational changes that will correspond to requirements of the new system. The group should not attempt to press or sell a reorganization. Major changes in organization are the prerogatives of top management.

Organization for systems management requires an outlook different from that of organization for functional component management. Subsystem managers should recognize that their main objective is that the subsystem should function in a way that is best for the whole system. Subsystem management requires a knowledge of the dynamics of systems and of the need for trade-offs to optimize system performance. The manager must be able to interface his or her subsystem operations effectively with coupled subsystems.

The hierarchy of management should follow the hierarchy of systems and subsystems rather than of technical disciplines. Assignment to activities by technical discipline should be primarily at the lower level, except for some overlay service systems such as financial planning and control.

Document The Detailed Design

The end of the detailed design project is production of the documents that specify the system, its operation, and its design justification. Documentation consists of

1. A summary flowchart.
2. Detailed flowcharts.
3. Operations activity sheets showing inputs, outputs, and transfer functions.
4. Specification of the database or master file.
5. Computer hardware requirements.
6. Software (programs).
7. Personnel requirements by type of skill or discipline.
8. Final (updated) performance specifications.
9. Cost of installation and implementation of the system.
10. Cost of operating the system per unit of time.
11. Program for modification or termination of the system.
12. An executive digest of the MIS design. This is a report that top management can read rapidly to get the essence of the system, its potential for the company, its cost, and its general configuration. We point out that a high-level MIS official at General Electric remarked, "If the MIS can be justified on the basis of cost savings, it isn't an MIS." The executive digest should be directed toward showing how the system will aid managers' decision making by gains in information or in time.

Some documentation should be on standardized forms. Input-output-activ-ity diagrams or listings are an example. Obviously, standard symbols should be used on flowcharts and guidelines should be established for flowchart format. Some documentation is unique to a project, such as the database, and the format and classification of items should be determined by the needs of the particular user. Other documentation should simply follow good reporting style.

Revisit The Manager-user

The system design is much firmer now and it is time to repeat the procedures of–

- Reporting the status to upper management
- Feedback to and request for support from lower-level management
- Education and gentle selling to non-management users.

Questions

Very Short Questions :

1. What is PERT diagram ?
2. What is task force meeting ?
3. What is internal source documents ?
4. What do you mean by subsystem ?

Short Questions :

1. Draw a hierarchy of activites of detailed design.
2. What type of documentation are needed of detailed design ?
3. Explain the aim of Detailed Design ?
4. What do you mean by project planning ?

Long Question :

1. What is Non-Technical and Technical Design ? Depoly all parts of Technical Design.
2. What is Detailed Design ? Discuss it's aim and Explain how can define subsystem by using Detailed Design ?
3. Explain all Principles or Technique for information capturing.
4. Short notes on : (a) Modeling (b) Simulation (c) Test Planning
5. What is the aim of Dutailed Design
6. What do you mean by Project Planning
7. What are the principal sources for abtaining information ?

❑❑❑

13

Client- Server Model of Ecommerce and High Level Design

E-commerce Architecture: E-commerce is based on the client-server architecture. A client can be an application, which uses a Graphical User Interface (GUI) that sends request to a server for certain services. The server is the provider of the services requested by the client. In E-commerce, a client refers to a customer who requests for certain services and the server refers to the business application through which the services are provided. The business application that provides services is deployed on a Web server. The Web server is a computer program that provides services to other computer programs and serves requested Hyper Text Mark-up Language (HTML) pages or files.

In client-server architecture, a machine can be both a client as well as a server. There are two types of client server architecture that E-commerce follows: two-tier and three-tier.

Two-Tier Architecture:

In two-tier client-server architecture the user interface runs on the client and the database is stored on the server. The business application logic can either run on the client or the server. The user application logic can either run on the client or the server. It allows the client processes to run separately from the server processes on different computers. The client processes provide an interface for the customer that gather and present the data on the computer of the customer. This part of the application is known as presentation layer. The server processes provide an interface with the data store of the business. This part of the application is known as data layer. The business logic, which validates data, monitors security and permissions and performs other business rules, can be kept either on the client or the server. The following Figure shows the outline of the two-tier architecture.

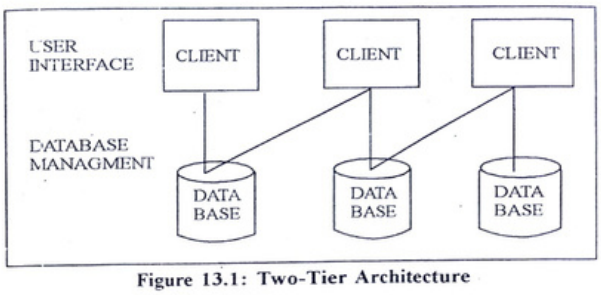

Figure 13.1: Two-Tier Architecture

Three-tier architecture:

The three-tier architecture emerged in the 1990s to overcome the limitations of the two-tier architecture. In three-tier architecture, the user interface and the business application logic, also known as business rules and data storage and access, are developed and maintained as independent modules. The three-tier architecture includes three tiers: top tier, middle tier and third tier. The top tier includes a user interface where user services such as session, text input, and dialog and display management reside. The middle tier provides process management services such as process development, process monitoring and process resourcing that are shared by the multiple applications. The third tier provides database management functionality. The data management component ensures that the data is consistent throughout the distributed environment; the centralized process logic in this architecture, which makes administration easier by localizing the system functionality, is placed on the middle tier. The following Figure shows the outline of the three-tier architecture.

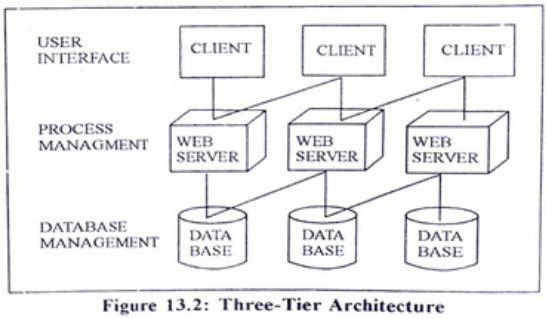

Figure 13.2: Three-Tier Architecture

The client server architecture offers many advantages which are as follows: The client-server architecture provides standardized, abstract interfaces to establish communication between multiple modules. When these modules are combined, they become an integrated business application. Each module is a shareable and reusable object that can be included in another business application. In the client-server architecture, the functions of a business application are isolated within the smaller business application objects and so application logic can be modified easily. In "the client-server architecture, each business application object works with its own encapsulated data structures that correspond to a specific database. When business application objects communicate, they send the data parameters as specified in the abstract interface rather than the entire database records. This reduces the network traffic. In the client-server architecture, a programmer can develop presentation components without knowing the business application logic. This architecture also helps a database analyst in accessing the data from the database without being concerned how the data is presented to an end user

High-level design

Introduction

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system. In contrast, low-level design further exposes the logical detailed design of each of these elements for programmers. High level design involves decomposing a system into modules, and representing the interfaces & invocation relationships among modules. A HLD is referred to as software architecture.

A high-level design provides an overview of a solution, platform, system, product, service or process.

Such an overview is important in a multiproject development to make sure that each supporting component design will be compatible with its neighbouring designs and with the big picture.

The highest-level solution design should briefly describe all platforms, systems, products, services and processes that it depends on and include any important changes that need to be made to them.

In addition, there should be brief consideration of all significant commercial, legal, environmental, security, safety and technical risks, issues and assumptions.

The idea is to mention every work area briefly, clearly delegating the ownership of more detailed design activity whilst also encouraging effective collaboration between the various project teams.

Today, most high-level designs require contributions from a number of experts, representing many distinct professional disciplines.

Finally, every type of end-user should be identified in the high-level design and each contributing design should give due consideration to customer experience.

Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level. This document also includes a high-level architecture diagram depicting the structure of the system, such as the database architecture, application architecture (layers), application flow (navigation), security architecture and technology architecture.

Scope of HLD

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

The HLD will:

- present all of the design aspects and define them in detail
- describe the user interface being implemented
- describe the hardware and software interfaces • describe the performance requirements
- include design features and the architecture of the project
- list and describe the non-functional attributes like: security, reliability, maintainability ,portability, reusability, application compatibility, resource utilization, serviceability

Performing High Level Design

Architectural Approach for Small and Medium sized Projects we stated that one of the key design documents for the project is the High Level Design document. This is the document that sets out the Conceptual and Logical views of the solution. The purpose is not only to describe the solution, but to show at a high level how the solution fits together and how it fits within your organisation. The organisational fit can be overlooked when people focus too much on the technology, particularly new technologies. It is also important to set out how the business will support a new solution e.g. are new skills and additional training required, to ensure the operational cost is appropriate.

Conceptual and Logical Views

The High Level Design sits at the Conceptual and the Logical levels of abstraction for a project. The diagrams and descriptions should focus at this level. The separation can be confusing to new architects. The simple rule I apply is that at the Conceptual level you do not have servers etc., but have objects which are familiar to the business people, such as buildings, people etc. At the Logical level you will have servers, workstations by their role, but do not show the multiples that are to be deployed to provide high availability.

To demonstrate this we will use a fictional new financial management solution for a business e.g. SAP or Oracle Financials, which is to have the safeguard of a standby site. The solution will have users at several sites performing different roles and accessing different types of backend system at a primary site. The standby site will provide disaster recovery capability and business continuance in the event of a serous outage or loss of service at the primary site.

The Conceptual view of this will show the locations for the people and both data centres, with explanation of what is performed at each. This will include the main roles performed by the users at the office locations. The Logical view will show the types of components to be used and the main technologies to be applied at these locations. Not simply Oracle or SAP but which parts of the products to provide which roles, e.g. General Ledger, Asset Management, Accounts Receivable, Reporting, etc. For the user workstations these are shown by the type of role they perform (Sales and Distribution, Financial Management, etc.) and grouped to indicate the different node types. The node type indicates that different software or hardware may be required. For example an Office Workstation may be the standard PC used within the business and capable of being used for a wide variety of the roles (S&D, FM, etc.). Additional special node types can be the Analysis workstation, Upload workstation, and BACS workstation that require additional software or hardware. If virtualisation technology is used the workstation nodes still apply as these indicate the different packaged builds for that type of role.

High Level Design Content

The High Level Design audience is both Technical and Business. This can be difficult as the business may not understand all of the technical aspects. However it is necessary to show that the main drivers and objectives have been addressed, and convey this in language that the business understands. It is also important to state early in the document were the solution is proposing a change of direct for the business, such as a move to a managed service or the use of a cloud service.

Some of the sub sections may not be appropriate for a particular solution. However I recommend that the heading remains and that a statement is made on why the section does not apply to that particular solution. This ensures that the overall content template does not reduce over time. Architects will use previous High Level Designs as a guide to the type of content to include.

- **Introduction**– The Introduction section sets out the broad boundaries of the solution at the Conceptual level of granularity. It is made up of the following sub sections:
- **Overview** – The Overview section sets out a general description of the solution, the main functionality to be provided, and the timeline for implementation. Where the design approach is different to other solutions for the business this is also set out in the overview. This reaffirms to the business and the stakeholders that you have understood what the aims are and provides a business context for the technical audience;
- **Requirements** – The requirements section should only list the few main requirements for the solution. These will include business requirements and any performance and volumes requirements that are key to the solution;
- **Constraints** – Most solutions are constrained in some way. This is a bullet list of the main constraints of the solution. These will include both business constraints and technical constraints. However they can also include legal and standards constraints;
- **Assumptions** – The solution will be based around a number of assumptions, such as the expected peak numbers of users, required level of availability, data growth requirements, access devices. This section provides a bulleted list of the main assumptions used for the solution;
- **Risks** – The project will operate a risk register for project risk and mitigation action. However the solution may use components that are new or new to the organisation. This section lists the solution risks. One of the optional items to add could be a sizing risk where the size modelling has been problematic. This type of risk is often seen for new solutions where the expected level of take up can vary over a wide range;
- **Key Drivers** –The drivers section includes both the key business and the technology drivers. The business drivers can include economic factors as well as changes within the business. The technology drivers will include new types of product, e.g. virtualisation, cloud opportunities, smartphone and tablet devices;
- **Principles** – What are the principles on which the solution is based? These will cover general principles such as the early adoption of promising technologies. However they must cover the Business, Data, Security and Technology principles.
- **Architectural View**– This section provides the Logical level and sets out two main operating models; the Current Operating Model (COM), and the Target Operating Model (TOM). These are described along with several other sub sections on specific solutions architecture aspects:

- **Current Operating Model** – The COM provides the current or As-Is view. In some instances there is no existing COM as this is a new idea for the business. However the COM section can cover how the concept is achieved currently elsewhere or the current alternatives;
- **Target Operating Model** – The TOM is the To-Be view, or the position at delivery of the solution. Where a solution is delivered in phases over a long period of time, it is appropriate to show the TOM for each phase. This provides the business with an understanding of the limitations and opportunities of the interim solution and the complete solution;
- **Analysis** – This section will provide a summary of the analysis work used to arrive at the sizing. Areas with a high volume need a separate metrics model which I will cover in a future article. This summary should include any profile of activity over the day, week, or month to show expected peaks and lows. For example end of month invoice volumes. It should also cover the breakdown of the types of things so that the mix is understood. Again this could be expenses for employees, invoices for customers, payments to suppliers. This allows variation for handling to be fully assessed in the solution and peak load activity such as end of year to be quantified. Very high end of year volumes may needs additional resources to be added for a few weeks, or a reduction of service to some less critical business functions. These need to be designed in from the outset to achieve the best fit and agreement from the organisation;
- **Availability** – The Availability section sets out the methods to be used in the solution to provide the level of uptime the business require. This will include a bulleted statement on the main small component considerations such as resilience from a failure of a server, network switch, user workstation, up to network circuit and data centre loss through UPS or environmental failure. The detail is part of the Detail Design, this section provides the main methods used in the solution to provide the required availability safeguards;
- **Security** – Most solutions have a need to enforce security. This section will use the security principles identified in the earlier part of the High Level Design to describe how these are to be enforced. This can include reference to specific technologies used for end users and interfacing systems, such as end to end encryption using SSL. However it will also cover support security controls, such as disabling accounts for support staff that leave or change role within the organisation;
- **Scalability** – The popularity of a service cannot always be determined accurately. Some solutions begin with a reduced size solution and invest in additional resources when the demand growth is proven. With commodity

hardware it is possible to add more servers (horizontally scale) or to replace the existing servers with more powerful servers (vertical scale). Vertical scaling can offer a better return on licensing as the number of cores used remains lower, but is usually more disruptive. This section addresses the option as part of the whole life solution. It ensures that when the solution does need to scale up (or down if it begins with an initial registration peak) is has been addressed as part of the solution and not added later;

- **Portability** – The portability of a solution can be considered for technology options. This is used to identify lock in to proprietary technologies that may not develop as the organisation expect. However I also use this section to cover how the data for the solution is migrated at the end of life of the hardware and the software. This allows mechanisms to be built in at the outset that make the migration easier;
- **Reporting** – Most solutions have some reporting requirement. Even where the business is not looking to provide a range of classic reports, there will be reporting of usage such as web analysis. This section is also used to set out how this information is made available to business. If there is an existing reporting service in the business, how will that access this new data to avoid the business ending up with multiple reporting solutions;
- **Solution Administration** – The administration section is aimed at the business administration of the solution not the support administration. Many solutions now have support of the key information by business areas. IT Support will look after the health of database, but a business area may be required to edit or correct invoice or payment details. Separating the administration of the business information from the IT support needs to be designed into the solution so that the appropriate roles and controls are in place.
- **Business Processes**–Following on from the Architectural View that shows the Current and the Target Operating Models, it is important to understand the business process changes. Not all solutions introduce new business processes or change the existing processes, however where they do the business need to agree these. The section is not intended to provide the low level detail of the business processes. However it should provide the main process categories (or service), the steps in the process, and the role that carries out each step. For example this could be the Create a Supplier Service that includes processes within the solution and outside, such as the Dun & Bradstreet or similar credit checks. This helps to identify roles to be catered for in the solution design and what authority is required. I find this section helps the business understand what the solution will mean to them. It gives an opportunity for early feedback on whether the level of responsibility is appropriate.

- **Deployment View**–The Deployment View in the High Level Design is focussed on the key aspects of the solution at the Logical level. This section is expanded in the Detail Design with the physical layout and component specification. Within the High Level design this describes what each of the components provides to the overall solutions. This is divided into the sub sections:
- **Business**–The Business section describes each business role within the solution. However it also covers any location changes i.e. more of fewer sites, and any organisational changes, such as a restructuring of sections or departments. The organisational and location changes can have a significant impact on the project timescales and may require specialist change management. Where these are identified late in the project life they can introduce delays or result in the organisation being unprepared.
- **Data**–The data section will use the outcome from the analysis to determine data volumes. However where the data cannot be held indefinitely or is to be moved to a lower tier of storage, the process to delete or archive needs to be identified. Some solutions leave the delete activity until required i.e. 4½ years in on the deletion of 5 year old data. However this produces considerable risk as the DBA's assigned the role are not as close to the fundamentals of the design as the original designers. The detail decisions, such as what to do with old reference data, are part of the Detailed Design. However the rationale for leaving this to later must be covered in this section of the High Level Design. This section also covers the data interface where the solution is to interface with other systems;
- **Application**–The application section is used to describe the capabilities that are being provided in the solution, such as General Ledger or BACS Payment. This is not the specific product features, which are covered in the Technology section. The business will use this section to help identify where some expected capabilities are missing or incomplete. This will ensure that any initial deficiencies in the solution are identified and resolved early;
- **Technology**–Technology is more than the servers, networks and workstations as it also includes the software products and choice of operating systems. Within this section I sub group these under the headings Application Software, Operating Systems, Networks, Servers, Workstations, and Peripherals. Describing within each group what is to be used and the role it performs in the solution. It is also worth listing where the component is new to the organisation. Solutions which also need development and test environments have smaller sub sections setting out each of the other environments e.g. Development, Test, and Pre-Production. For the other environments it is important to explain what it is used for and the limitation

of the environment. It is particularly important to recognise that IT Support also needs access to environments for trialling patch and service pack deployments. In a solution developed by in-house teams, with a split between infrastructure component and application component development. There is often a need for separate development environments. In ensuring that all environments are covered in the design the full cost to the business will be understood;

- **Support**–The support section provides the IT support elements off the solution at a high level. For mature organisations this will build of current support and monitoring practises. However these practises need to be listed so there is a clear statement of what is required. This will highlight where the solution will require training for existing support staff, and whether an increase in support numbers is required. If the solution introduces a need for a change in the support model from a daytime Monday to Friday support to 7 days, or even a 24 hours support model, it is set out here. The support section also needs to address the expected timeline for adoption by the support team (to allow for training and knowledge transfer), and whether initial support is shared with the development team in the early months.

High Level Diagram Template for applying HLD

- Cover Page
- Table of Contents
- Introduction
- Problem Statement and Proposed Solution

This section describes the problem and how you intend to solve it.

- System Description and Block Diagram

  Give a block diagram of your system, and describe all of the major blocks and major interfaces between blocks.
- System Requirements

Overall System:

Give the overall system requirements. These requirements should describe what your system does (and what will be demonstrated in May.) There are detailed subsystem descriptions in the subsequent section.

Subsystem Requirements:

The requirements of each subsystem or major interface are described here by subsystem. These lower level requirements support the overall system requirements. Note that major interfaces (such as a wireless interface) should be described like any other subsystem. Don't forget that there will be software as well has hardware in many of the subsystems, and that software will have requirements.

Future Enhancement Requirements

There may be a number of features that aren't going to be part of the initial release of your product, but that you would like to add in the future. These are listed here so that the design does not preclude adding these features.

- High Level Design Decisions

Broken down by subsystem and major interface, this section presents your high level design of each subsystem or interface.

For each subsystem or major interface, you should describe the function or interface and the appropriate technologies that will go into the subsystem. The decision level here is not necessarily to the specific part, but rather to a technology.

For example, if your system is using a wireless interface, your design should include the choices you considered for the wireless interface, the technology you chose to use, and why that technology choice was made (requirement such as size, cost, speed, power, etc. might all be issues for a wireless technology.) The specific part that you will be using can be left to the low level design.

In like fashion, if a subsystem contains embedded intelligence, it is not necessary to specify the specific microcontroller that you will be using. The requirements listed earlier should allow you to specify a class of microcontrollers (based again on requirements like cost, power (electrical), power (processing power), I/O and interface requirements, etc.)

- Major Component Costs

Specify the costs of your major system components. This should allow you to have a rough cost estimate of your system.

- Conclusions

Questions

Very Short Questions :

1. What do you mean by clint ?
2. What do you mean by tirver ?
3. HLD Stand for ?
4. What is key Drivers ?
5. What is COM ?
6. What is TOM ?

Short Questions :

1. What is two Tier Architecture ?
2. What is the Scope of HLD ?
3. What is the difference between conceptual and Loical views ?
4. Explain the roll of HLD ?

Long Question :

1. What is the difference between Two-Tier or Three Tier Architecture ?
2. What do you meen by High Level Design? Explain it is conceptual and logical view of High Level Desing.
3. Breafly explain High Llevel Design Contonts-
4. What is Client-Server System ? Explain all it is characterstics ?

❑❑❑