Engineering

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.

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 of Web E-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 nontechnical members only.

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

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.

In order to perform technical design activity effectively, there are four technical elements which a Web engineer should work to reuse. These technical 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. 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, Powell 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. 17.1.

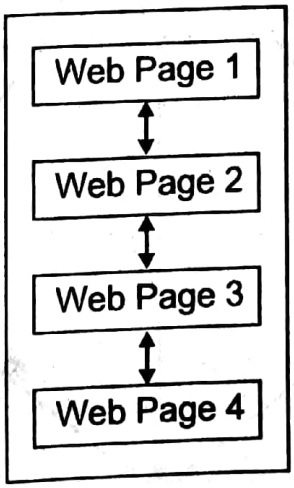

Fig. 17.1 : Linear Structure – Simple Linear

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. 17.2

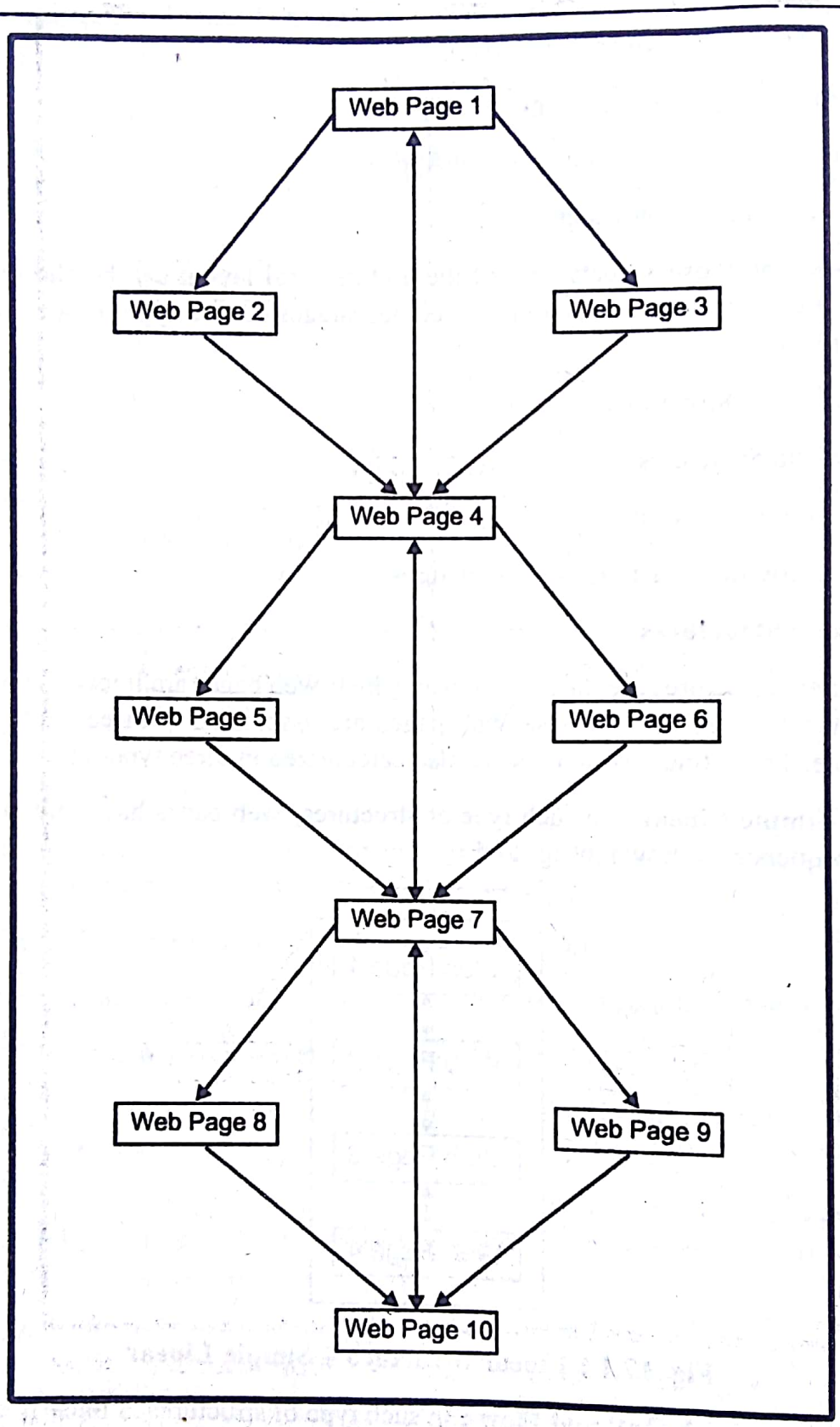

Fig. 17.2 : Linear Structure – Linear with Optional Flow

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. 17.3

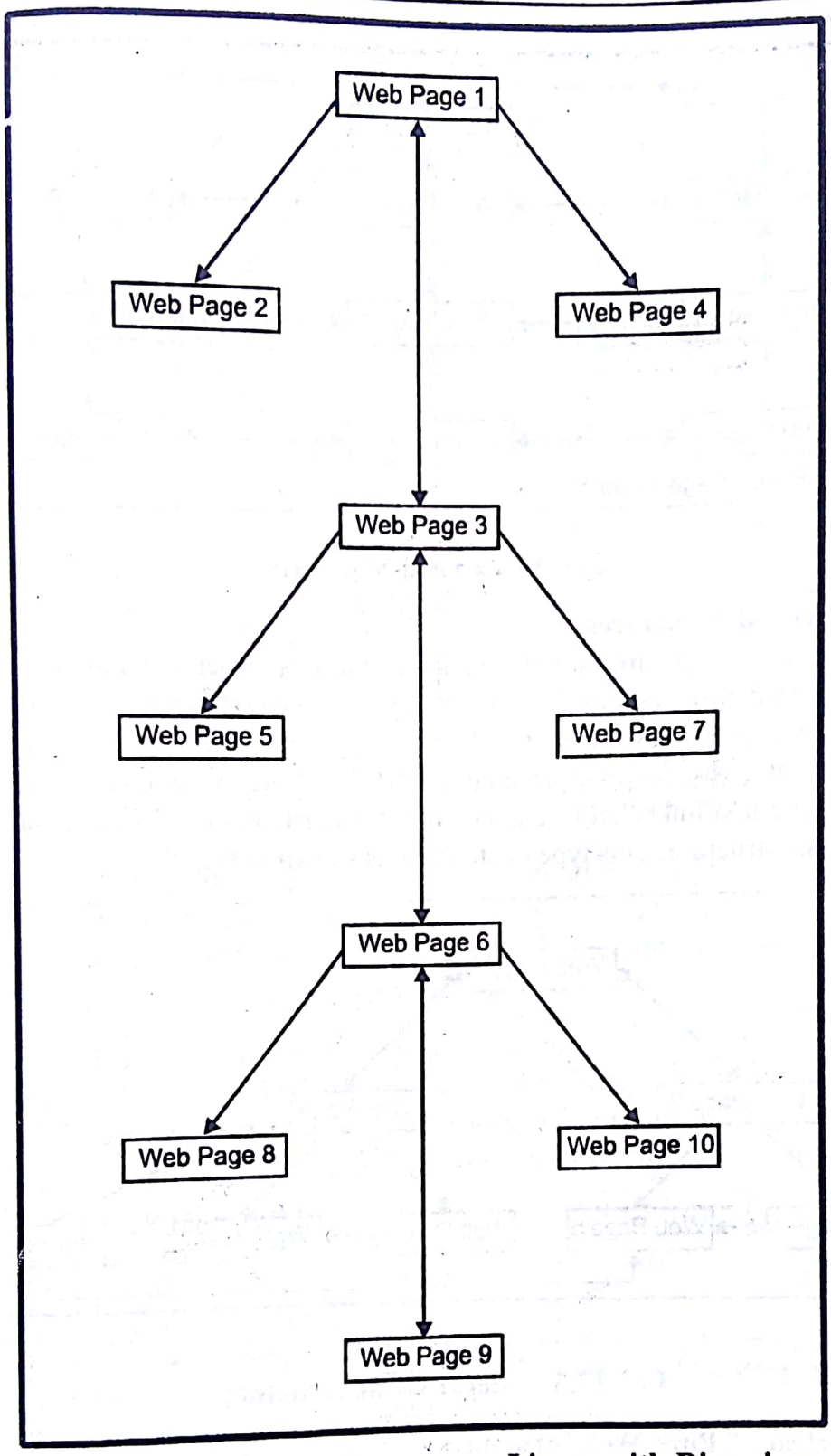

Fig. 17.3 : Linear Structure – Linear with Diversions

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. 17.4.

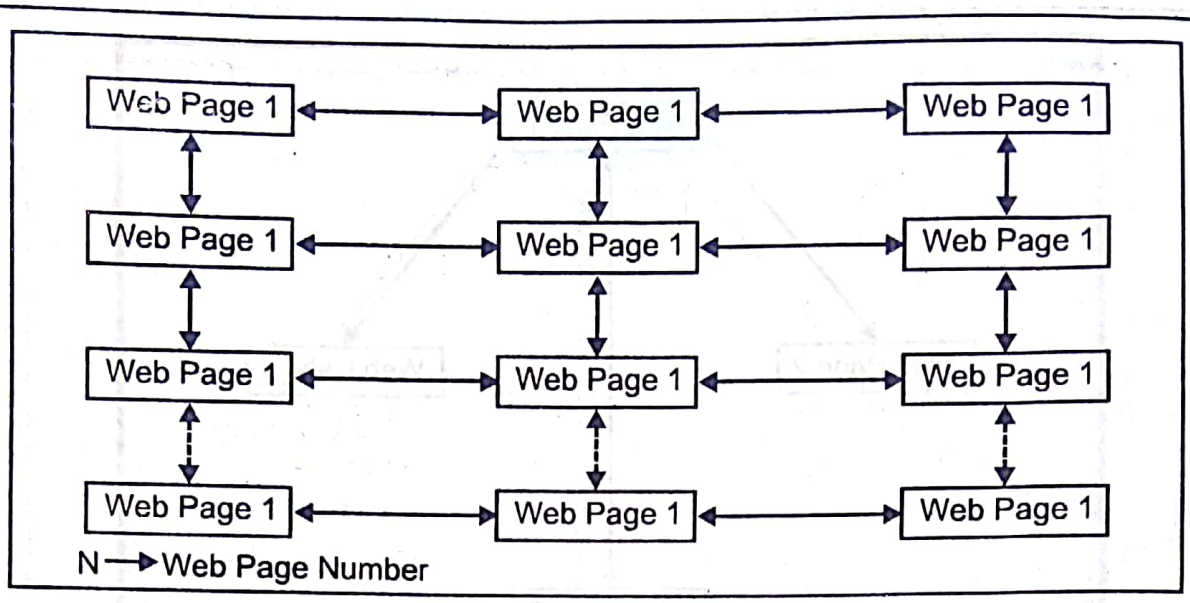

Fig. 17.4 : Grid Structure

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. 17.5.

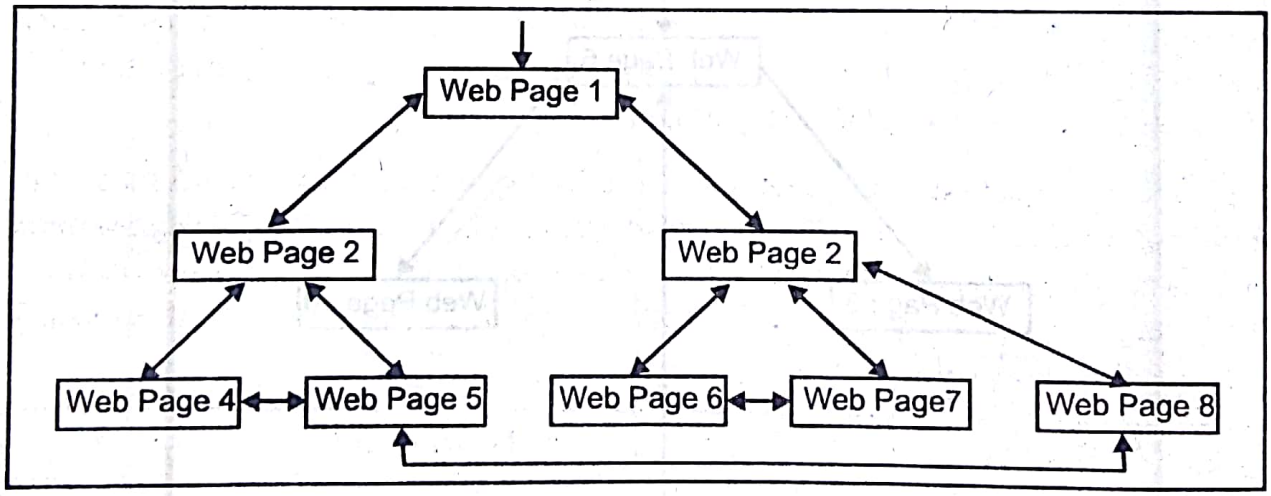

Fig. 17.5 : Hierarchical Structure

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. 17.6.

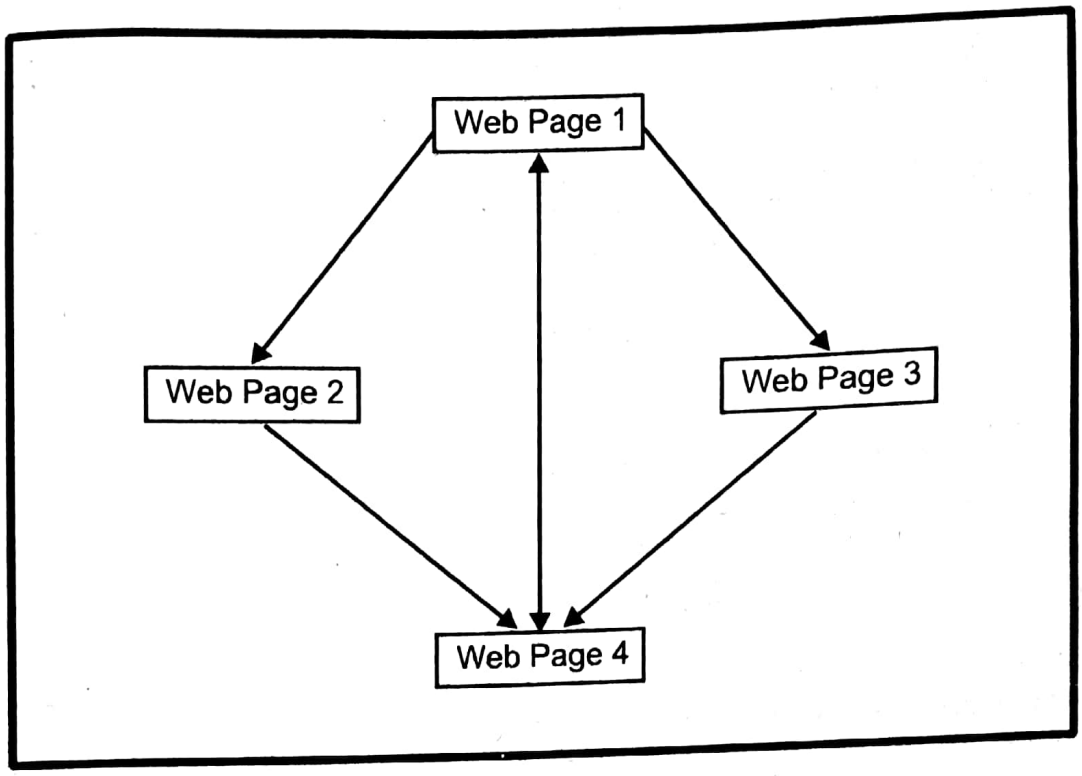

Fig. 17.6 : Networked or Pure Web Structures

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-1

S. No.	Desing Pattern	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.

Table 17.7 : Various Design Patterns and Their Description

(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.

DETAILED SYSTEM DESIGN

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

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 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 SYSTEM, the detailed design yields the construction and operating specifications.

PROJECT MANAGEMENT OF SYSTEM 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 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 system project and the cost of developing the detailed design of the project. The key steps in planning and control of detailed design.

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 system 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.

HIGH LEVEL DESIGN

High Level Design means precisely that. A high level design discusses an overall view of how something should work and the top-level components that will comprise the proposed solution. It should have very little detail on implementation, i.e. no explicit class definitions, and in some cases not even details such as database type (relational or object) and programming language and platform. A low level design has nuts and bolts type detail in it which must come after high level design has been signed off by the users, as the high level design is much easier to change than the low level design.

Software sometimes can be a complex entity. Its development usually follows what is known as **Software Development Life Cycle** (SDLC). High Level Design is one of the initial stages.

Typically a Software Development Life-cycle (SDLC) has four stages-

- Requirements
- Design
- Coding
- Implementation

The first stage involves requirement analysis and mutual understanding of the requirements between the consumer and the developer.

The second stage in the SDLC is the Design Stage. The objective of the design stage is to produce the overall design of the software. The design stage involves two sub-stages namely :

- ✓ High-Level Design
- ✓ Detailed-Level Design

In the High-Level Design, the Technical Architect of the project will study the proposed applications functional and non-functional (qualitative) requirements and design an overall solution architecture of the application which can handle those needs.

The High-Level Design has the following activities :

- ➢ Design of the Solution Architecture for the project
- ➢ Design of the technical Architecture for the project
- ➢ Development of the Logical model (e.g Entity Relation Diagram or Object Model) for the project
- ➢ Development of the Physical Data Model for the project

High Level Design is the first of a (possibly) two-stage design process. The purpose of High Level Design is to:

- Make as many design decisions as possible before investing in coding effort
- Incorporate feedback from users and managers
- Provide documentation

The audience of High Level Design includes the customer, managers, and users.

Key Points

There are many, many points that can be made about correct approach to design. However, in general, Software Partners believes that good design will:

- Flow with the software tool or platform that is being used.

It is generally best not to try to force the tool into a preconceived notion of how the final application should look. Usually, it is possible to find an equally good or better solution to the business need that capitalizes on the way the software tool already works.

- o Focus on the business needs. Often, a more efficient solution is "better" than a fancier or more ingenious solution. It doesn't have to be brilliant - it does have to work.
- o Provide solutions to the key technical challenges of the project.

Deliverables

This process results in a Design Review and (usually) a High Level Design Document. The High Level Design document is medium to long, includes some dia-

grams and pictures, and is supplemented with software objects such as screen prototypes. It may show:

- Screen prototypes
- Narrative description of screen functions
- Data Model
- Major software module identification and description
- Interface definitions
- How key challenges and problems will be solved (example: how a simulation will be modeled using events and a concept of time).

What is 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.

Scope

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