UNIT 3 SOFTWARE DESIGN

Structure **Page Nos.**

3.0 INTRODUCTION

Software design is all about developing blue print for designing workable software. The goal of software designer is to develop model that can be translated into software. Unlike design in civil and mechanical engineering, software design is new and evolving discipline contrary classical building design etc. In early days, software development mostly concentrated on writing code. Software design is central to software engineering process. Various design models are developed during design phase. The design models are further refined to develop detailed design models which are closely related to the program.

3. 1 OBJECTIVES

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

- design data;
- understand architectural design;
- develop modular design, and
- know the significance of Human Computer Interface.

3.2 DATA DESIGN

To learn the process of system design involved to translate user requirement to implementable models.

- Designing of Data
- Architectural design which gives a holistic architecture of the software product
- Design of cohesive and loosely coupled module
- Design of interface and tips for good interface design
- Design of Human interface

Software Design Process

Software design is the process of applying various software engineering techniques to develop models to define a software system, which provides sufficient details for actual realisation of the software. The goal of software design is to translate user requirements into an implementable program.

Software design is the only way through which we can translate user requirements to workable software. In contrary to designing a building software design is not a fully developed and matured process. Nevertheless the techniques available provides us tools for a systematic approach to the design of software. *Figure 3.1* depicts the process of software design.

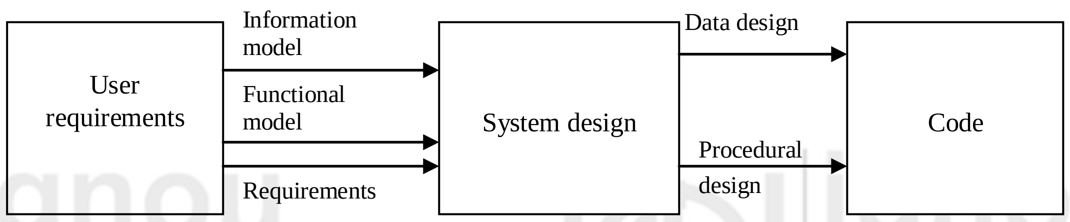

Figure 3.1 : Process of software Design

During the process of software design, the information model is translated to data design. Functional model and behavioural model are translated to architectural design, which defines major component of the software. Keeping in view of the importance of design, it should be given due weightage before rushing to the coding of software.

Software design forms the foundation for implementation of a software system and helps in the maintenance of software in future too. Software quality and software design process are highly interrelated. Quality is built into the software during the design phase.

High level design gives a holistic view of the software to be built, where as low level refinements of the design are very closely related to the final source code. A good design can make the work of programmer easy and hardly allow the programmer to forget the required details. Sufficient time should be devoted to design process to ensure good quality software.

The following are some of the fundamentals of design:

- The design should follow a hierarchical organisation.
- Design should be modular which are logically partitioned into modules which are relatively independent and perform independent task.
- Design leading to interface that facilitate interaction with external environment.
- Step wise refinement to more detailed design which provides necessary details for the developer of code.
- Modularity is encouraged to facilitate parallel development, but, at the same time, too many modules lead to the increase of effort involved in integrating the modules.

Data design is the first and the foremost activity of system Design. Before going into the details of data design, let us discuss what is data? Data describes a real-world information resource that is important for the application. Data describes various entities like customer, people, asset, student records etc.

Identifying data in system design is an iterative process. At the highest level, data is defined in a very vague manner. A high level design describes how the application handles these information resources. As we go into more details, we focus more on the types of data and it's properties. As you keep expanding the application to the business needs or business processes, we tend to focus more on the details.

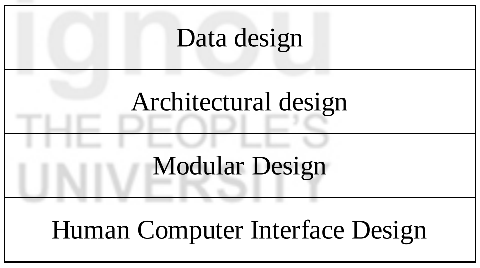

Figure 3.2 : Technical aspects of design

The primary objective of data design is to select logical representation of data items identified in requirement analysis phase. *Figure 3.2* depicts the technical aspects of design. As we begin documenting the data requirements for the application, the description for each item of data typically include the following:

- Name of the data item
- General description of the data item
- Characteristics of data item
- Ownership of the data item
- Logical events, processes, and relationships.

Data Sructure

A data structure defines a logical relationship between individual elements of a group of data. We must understand the fact that the structure of information affects the design of procedures/algorithms. Proper selection of data structure is very important in data designing. Simple data structure like a scalar item forms the building block of more sophisticated and complex data structures.

A scalar item is the simplest form of data. For example, January (Name of the Month), where as the collection of months in a year form a data structure called a vector item.

Example:

Month as string

months_in_year = [Jan, …, Dec]

The items in the vector called array is sequenced and index in a manner so as to retive particular element in the arry.

Month_in_year[4] will retrieve Apr

The vector items are in contiguous memory locations in the computer memory. There are other variety of lists which are non-contiguously stored. These are called linked lists. There are other types of data structures known as hierarchical data structure such as tree. Trees are implemented through linked lists.

☞ Check Your Progress 1

1) Quality is built into the software during the design phase explain?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2) Structure of information effects the design of algorithm. True ☐ False ☐

3) Design form the foundation of Software Engineering, Explain?

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

3.3 ARCHITECTURAL DESIGN

The objective of architectural design is to develop a model of software architecture which gives a overall organisation of program module in the software product. Software architecture encompasses two aspects of structure of the data and hierarchical structure of the software components. Let us see how a single problem can be translated to a collection of solution domains (refer to *Figure 3.3*).

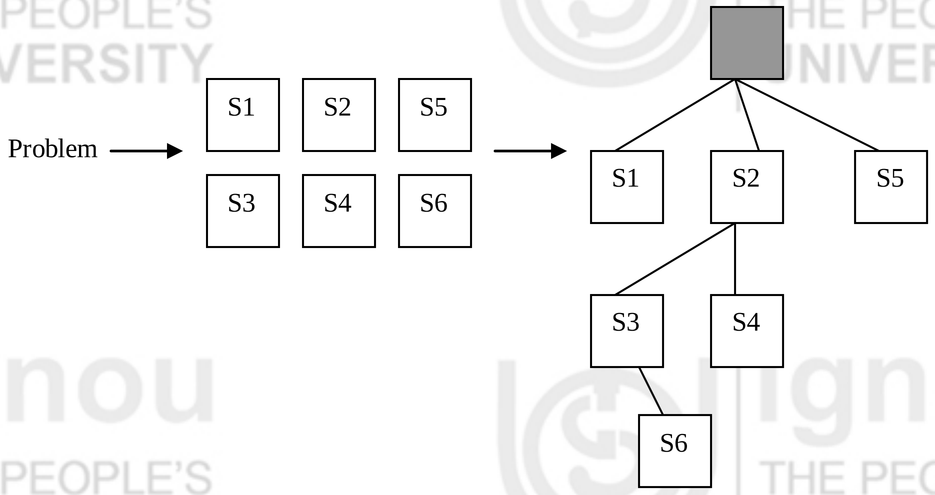

Figure 3.3: Problem, Solutions and Architecture

Architectural design defines organisation of program components. It does not provide the details of each components and its implementation. *Figure 3.4* depicts the architecture of a Financial Accounting System.

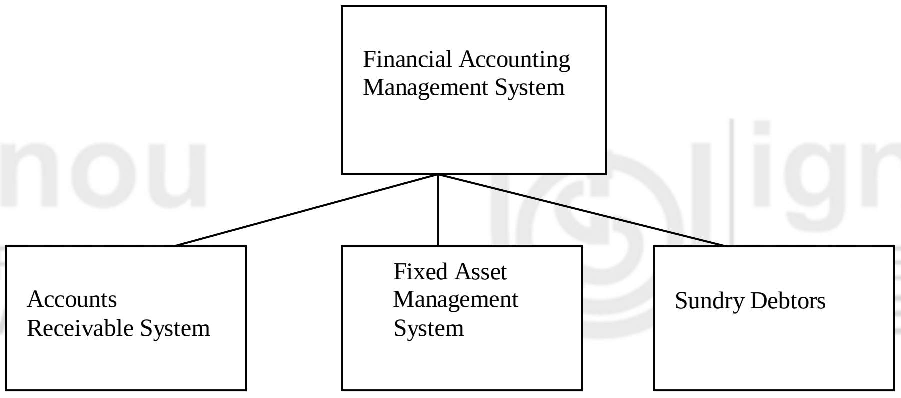
Financial Accounting Management System
Accounts Receivable System
Fixed Asset Management System
Sundry Debtors

Figure 3.4: Architecture of a financial accounting system

The objective of architectural design is also to control relationship between modules. One module may control another module or may be controlled by another module. These characteristics are defined by the fan-in and fan-out of a particular module. The organisation of module can be represented through a tree like structure.

Let us consider the following architecture of a software system.

The number of level of component in the structure is called depth and the number of component across the horizontal section is called width. The number of components which controls a said component is called fan-in i.e., the number of incoming edges to a component. The number of components that are controlled by the module is called fan-out i.e., the number of outgoing edges.

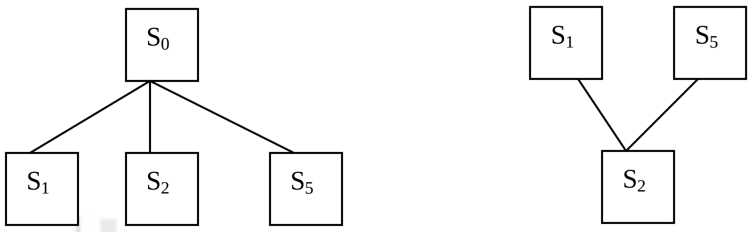

Figure 3.5: Fan-in and Fan-out

S_0 controls three components, hence the fan-out is 3. S_2 is controlled by two components, namely, S_1 and S_2, hence the fan-in is 2 (refer to *Figure 3.5*).

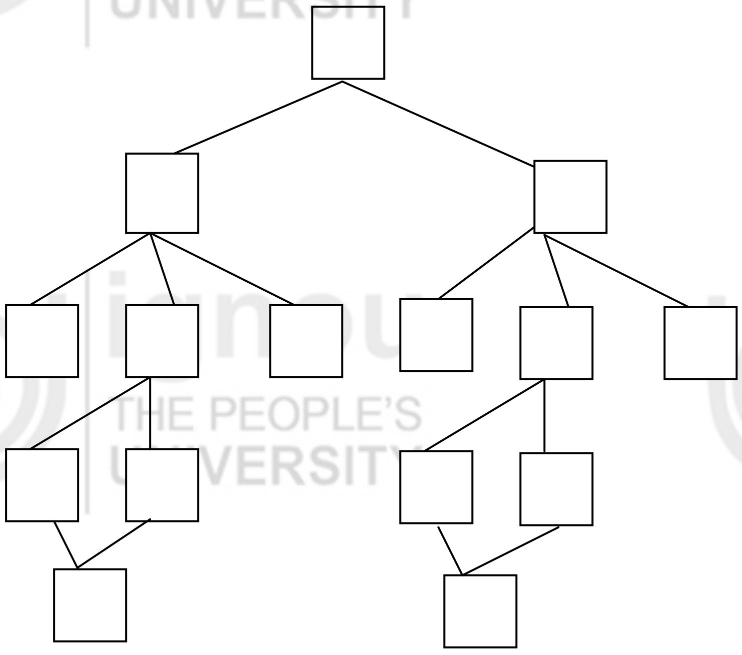

Figure 3.6 : Typical architecture of a system

The architectural design provides holistic picture of the software architecture and connectivity between different program components (refer to *Figure 3.6*).

3.4 MODULAR DESIGN

Modular design facilitates future maintenance of software. Modular design have become well accepted approach to built software product.

Software can be divided in to relatively independent, named and addressable component called a module. Modularity is the only attribute of a software product that makes it manageable and maintainable. The concept of modular approach has been derived from the fundamental concept of “divide and conquer”. Dividing the software to modules helps software developer to work on independent modules that can be later integrated to build the final product. In a sense, it encourages parallel development effort thus saving time and cost.

During designing of software one must keep a balance between number of modules and integration cost. Although, more numbers of modules make the software product more manageable and maintainable at the same time, as the number of modules increases, the cost of integrating the modules also increases.

Modules are generally activated by a reference or through a external system interrupt. Activation starts life of an module. A module can be classified three types depending activation mechanism.

- An incremental module is activated by an interruption and can be interrupted by another interrupt during the execution prior to completion.

- A sequential module is a module that is referenced by another module and without interruption of any external software.

- Parallel module are executed in parallel with another module

Sequential module is most common type of software module. While modularity is good for software quality, independence between various module are even better for software quality and manageability. Independence is a measured by two parameters called cohesion and coupling.

Cohesion is a measure of functional strength of a software module. This is the degree of interaction between statements in a module. Highly cohesive system requires little interaction with external module.

Coupling is a measure of interdependence between/among modules. This is a measure of degree of interaction between module i.e., their inter relationship.

Hence, highly cohesive modules are desirable. But, highly coupled modules are undesirable (refer to *Figure 3.7* and *Figure 3.8*).

Cohesion

Cohesiveness measure functional relationship of elements in a module. An element could be a instruction, a group of instructions, data definitions or reference to another module.

Cohesion tells us how efficiently we have positioned our system to modules. It may be noted that modules with good cohesion requires minimum coupling with other module.

Low cohesion
High cohesion
Undesirable
Desirable

Figure 3.7 : Cohesion

There are several types of cohesion arranged from bad to best.

- **Coincidental :** This is worst form of cohesion, where a module performs a number of unrelated task.
- **Logical :** Modules perform series of action, but are selected by a calling module.

- **Procedural :** Modules perform a series of steps. The elements in the module must takeup single control sequence and must be executed in a specific order.

- **Communicational :** All elements in the module is executed on the same data set and produces same output data set.

- **Sequential :** Output from one element is input to some other element in the module. Such modules operates on same data structure.

- **Functional :** Modules contain elements that perform exactly one function.

  The following are the disadvantages of low cohesion:

  - Difficult to maintain
  - Tends to depend on other module to perform certain tasks
  - Difficult to understand.

Coupling

In computer science, coupling is defined as degree to which a module interacts and communicates with another module to perform certain task. If one module relies on another the coupling is said to be high. Low level of coupling means a module doses not have to get concerned with the internal details of another module and interact with another module only with a suitable interface.

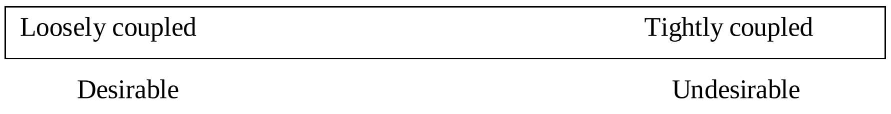

Figure 3.8 : Coupling

The types of coupling from best (lowest level of coupling) to worst (high level of coupling) are described below:

- **Data coupling:** Modules interact through parameters.
  Module X passes parameter A to module Y

- **Stamp coupling:** Modules shares composite data structure.

- **Control coupling:** One module control logic flow of another module. For example, passing a flag to another module which determines the sequence of action to be performed in the other module depending on the value of flag such as true or false.

- **External coupling:** Modules shares external data format. Mostly used in communication protocols and device interfaces

- **Common coupling:** Modules shares the same global data.

- **Content coupling:** One module modifies the data of another module.

Coupling and cohesion are in contrast to each other. High cohesion often correlates with low coupling and vice versa.

In computer science, we strive for low-coupling and high cohesive modules.
The following are the disadvantages of high coupling:

- Ripple effect.
- Difficult to reuse. Dependent modules must be included.
- Difficult to understand the function of a module in isolation.

Figure 3.9 depicts highly cohesive and loosely coupled system. *Figure 3.10* depicts a low cohesive and tightly coupled system.

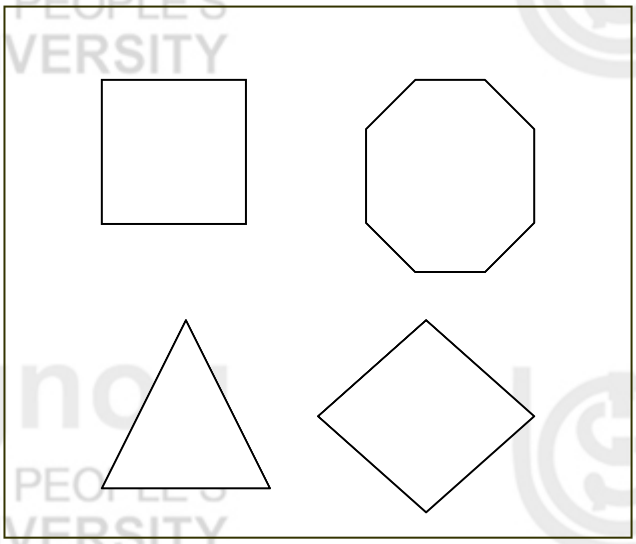

Figure 3.9 : High cohesive and loosely coupled system (desirable)

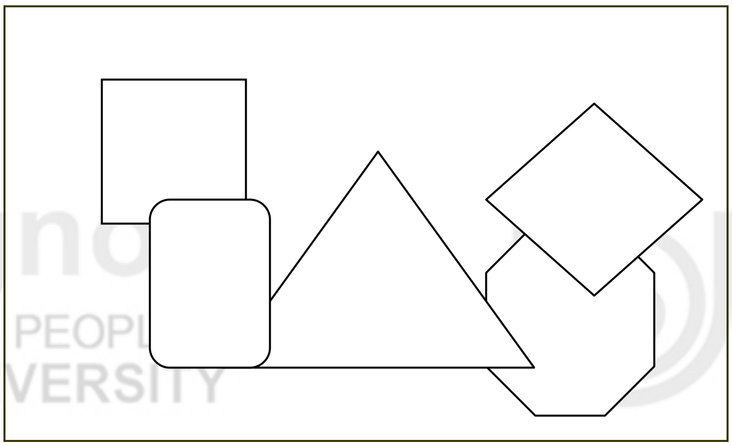

Figure 3.10 : Low cohesive and tightly coupled system (undesirable)

☞ **Check Your Progress 2**

1) Content coupling is a low level coupling True ☐ False ☐

2) Modules sharing global data refer to ____________.

3) Which is the best form of cohesion?

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

3.5 INTERFACE DESIGN

Interface design deals with the personnel issues of the software. Interface design is one of the most important part of software design. It is crucial in a sense that user interaction with the system takes place through various interfaces provided by the software product.

Think of the days of text based system where user had to type command on the command line to execute a simple task.

Example of a command line interface:

- run prog1.exe /i=2 message=on

The above command line interface executes a program prog1.exe with a input i=2 with message during execution set to on. Although such command line interface gives liberty to the user to run a program with a concise command. It is difficult for a novice user and is error prone. This also requires the user to remember the command for executing various commands with various details of options as shown above.

Example of Menu with option being asked from the user (refer to Figure 3.11).

```
To run the program select the
option below

    1.  Option 1
    2.  Option 2
    3.  Option 3
    4.  Back
    5.  Exit program

Enter your option ____
```

Figure 3.11 : A menu based interface

This simple menu allow the user to execute the program with option available as a selection and further have option for exiting the program and going back to previous screen. Although it provide grater flexibility than command line option and does not need the user to remember the command still user can't navigate to the desired option from this screen. At best user can go back to the previous screen to select a different option.

Modern graphical user interface provides tools for easy navigation and interactivity to the user to perform different tasks.

The following are the advantages of a Graphical User Interface (GUI):

- Various information can be display and allow user to switch to different task directly from the present screen.
- Useful graphical icons and pull down menu reduces typing effort by the user.
- Provides key-board shortcut to perform frequently performed tasks.
- Simultaneous operations of various task without loosing the present context.

Any interface design is targeted to users of different categories.

- Expert user with adequate knowledge of the system and application
- Average user with reasonable knowledge
- Novice user with little or no knowledge.

The following are the elements of good interface design:

- Goal and the intension of task must be identified.

- The important thing about designing interfaces is all about maintaining consistency. Use of consistent color scheme, message and terminologies helps.
- Develop standards for good interface design and stick to it.
- Use icons where ever possible to provide appropriate message.
- Allow user to undo the current command. This helps in undoing mistake committed by the user.
- Provide context sensitive help to guide the user.
- Use proper navigational scheme for easy navigation within the application.
- Discuss with the current user to improve the interface.
- Think from user prospective.
- The text appearing on the screen are primary source of information exchange between the user and the system. Avoid using abbreviation. Be very specific in communicating the mistake to the user. If possible provide the reason for error.
- Navigation within the screen is important and is specially useful for data entry screen where keyboard is used intensively to input data.
- Use of color should be of secondary importance. It may be kept in mind about user accessing application in a monochrome screen.
- Expect the user to make mistake and provide appropriate measure to handle such errors through proper interface design.
- Grouping of data element is important. Group related data items accordingly.
- Justify the data items.
- Avoid high density screen layout. Keep significant amount of screen blank.
- Make sure an accidental double click instead of a single click may does some thing unexpected.
- Provide file browser. Do not expect the user to remember the path of the required file.
- Provide key-board shortcut for frequently done tasks. This saves time.
- Provide on-line manual to help user in operating the software.
- Always allow a way out (i.e., cancellation of action already completed).
- Warn user about critical task, like deletion of file, updating of critical information.
- Programmers are not always good interface designer. Take help of expert professional who understands human perception better than programmers.
- Include all possible features in the application even if the feature is available in the operating system.

- Word the message carefully in a user understandable manner.
- Develop navigational procedure prior to developing the user interface.

3.6 DESIGN OF HUMAN COMPUTER INTERFACE

Human Computer Interface (HCI) design is topic that is gaining significance as the use of computers is growing. Let us look at a personal desktop PC that has following interface for humans to interact.

- A key board
- A Computer Mouse
- A Touch Screen (if equipped with)
- A program on your Windows machine that includes a trash can, icons of various programs, disk drives, and folders

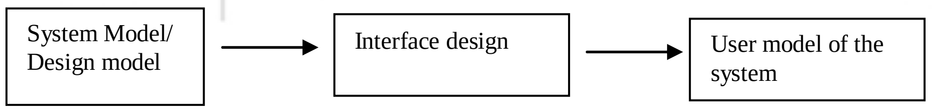

Figure 3.12: The process of interface design

What do these things have in common? These are all human-computer interfaces which were designed to make it easier to accomplish things with a computer. If we recall the early days computer, some one had to remember long cryptic strings of commands in accomplish simplest to simplest thing with a computer like coping a file from one folder to another or deleting a file etc. It is due to the evolution of human-computer interface and human-computer interface designers that's now being accomplished with the click of a mouse.

The overall process of design leads to the translation of design models to user model. Human-Computer Interface design goal is to discover the most efficient way to design interfaces for human interaction and understandable electronic messages. A lot of research is taking place in this area that even helps physically disabled person to operate computer as easily as a normal person. *Figure 3.12* depicts the process of interface design.

A branch of computer science is devoted to this area, with recommendations for the proper design of menus, icons, forms, messages, dialogue as well as data display. The user friendliness of the program we are using is a result of interface design. The buttons, pull down menu, context sensitive help have been designed to make it easy for you to access the program.

The process of HCI design begins with the a creation of a model of system function as perceived by the end user. These models are designed by delineating them from design issues and program structure. HCI establishes a user model of the overall system.

The following are some of the principles of Good Human computer interface design:

Diversity: Consider the types of user frequently use the system. The designer must consider the types of users ranging from novice user, knowledgeable but intermittent user and expert frequent user. Accommodating expectation of all types of user is important. Each type of user expects the screen layout to accommodate their desires, novices needing extensive help where as expert user want to accomplish the task in quickest possible time.

For example providing a command such as ^P (control P) to print a specific report as well as a printer icon to do the same job for the novice user.

Rules for Human Computer Interface Design:

1. **Consistency :**
    - Interface is deigned so as to ensure consistent sequences of actions for similar situations. Terminology should be used in prompts, menus, and help screens should be identical. Color scheme, layout and fonts should be consistently applied throughout the system.
2. **Enable expert users to use shortcuts:**
    - Use of short cut increases productivity. Short cut to increase the pace of interaction with use of, special keys and hidden commands.
3. **Informative feedback :**
    - The feedback should be informative and clear.
4. **Error prevention and handling common errors :**
    - Screen design should be such that users are unlikely to make a serious error. Highlight only actions relevant to current context. Allowing user to select options rather than filling up details. Don't allow alphabetic characters in numeric fields
    - In case of error, it should allow user to undo and offer simple, constructive, and specific instructions for recovery.

5 **Allow reversal of action :** Allow user to reverse action committed. Allow user to migrate to previous screen.

6. **Reduce effort of memorisation by the user :**
    - Do not expect user to remember information. A human mind can remember little information in short term memory. Reduce short term memory load by designing screens by providing options clearly using pull-down menus and icons
7. **Relevance of information :** The information displayed should be relevant to the present context of performing certain task.
8. **Screen size:** Consideration for screen size available to display the information. Try to accommodate selected information in case of limited size of the window.
9. **Minimize data input action :** Wherever possible, provide predefined selectable data inputs.
10. **Help :** Provide help for all input actions explaining details about the type of input expected by the system with example.

About Errors: Some errors are preventable. Prevent errors whenever possible. Steps can be taken so that errors are less likely to occur, using methods such as organising screens and menus functionally, designing screens to be distinctive and making it difficult for users to commit irreversible actions. Expect users to make errors, try to anticipate where they will go wrong and design with those actions in mind.

Norman's Research

Norman D. A. has contributed extensively to the field of human-computer interface design. This psychologist has taken insights from the field of industrial product design and applied them to the design of user interfaces.

According to Norman, design should use both knowledge in the world and knowledge in the head. Knowledge in the world is overt–we don't have to overload our short term memory by having to remember too many things (icons, buttons and menus provide us with knowledge in the world. The following are the mappings to be done:

- Mapping between actions and the resulting effect.
- Mapping between the information that is visible and the interpretation of the system state. For example, it should be obvious what the function of a button or menu is. Do not try to built your own meaning to the commonly used icons instead use conventions already established for it. Never use a search icon to print a page.
- Hyperlink should be used to migrate from one page to connected page.

☞ **Check Your Progress 3**

1) Keyboard short-cut is used to perform __________.

2) What are the types of user errors you may anticipate while designing a user interface. Explain?

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

3.7 SUMMARY

Design is a process of translating analysis model to design models that are further refined to produce detailed design models. The process of refinement is the process of elaboration to provides necessary details to the programmer. Data design deals with data structure selection and design. Modularity of program increases maintainability and encourages parallel development. The aim of good modular design is to produce highly cohesive and loosely coupled modules. Independence among modules is central to modularity. Good user interface design helps software to interact effectively to external environment. Tips for good interface design helps designer to achieve effective user interface.

Quality is built into software during the design of software. The final word is: Design process should be given due weightage before rushing for coding.

3.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) In building a design, where the quality of structural strength and other aspects are determined by the design of the building. In software design, the design process goes through a series of refinement process and reviews, to see whether the user requirements and other quality parameters are properly not reflected in the design model. A good design is more likely to deliver good software. The process of getting it right starts from the software design stage.

   - Data design affects the efficiency of algorithms.
   - Modularity enables the software component to perform distinct independent tasks with little chance of propagating errors to other module.
   - Good interface design minimizes user errors and increases user friendliness of the system.

2) True.

3) Software design is the first step of translating user requirement to technical domain of software development. Without designing there is always a risk of designing an unstable system. Without design we can't review the requirements of the product until it is delivered to the user. Design helps in future maintenance of the system.

Check Your Progress 2

1) False
2) Common coupling
3) Functional.

Check your Progress 3

1) Frequently performed tasks.

2) Anticipation user error provides vital input for user interface design. Prevent errors wherever possible and steps can be taken to design interface such that errors are less likely to occur. The methods that may be adopted may include well organisation of screen and menus functionally. Using user understandable language, designing screens to be distinctive and making it difficult for users to commit irreversible actions. Provide user confirmation to critical actions. Anticipate where user can go wrong and design the interface keeping this in mind.

3.9 FURTHER READINGS

1) *Software Engineering, Sixth Edition, 2001*, Ian Sommerville; Pearson Education.
2) *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill International Edition.
3) *Software Design : From Programming to Architecture, First Eition, 2003, Eric J. Braude; Wiley*

Reference Websites

http://www.ieee.org
http://www.rspa.com
http://sdg.csail.mit.edu