CHAPTER 2

FEATURES OF THE DESIGNED MODEL

In this chapter, highlights of work, which has been carried out on the topic of HCI, has been presented. We have started with importance of windows in advanced user interface. They allow the user to interact with multiple sources of information at the same time. Different video attributes underlining, inverse video and flashing have been discussed in subsequent section. Menu selection option and an improved technique for menu navigation are the element of next section. For the convenience of users, an approach to command name abbreviation has been adopted and has been found suitable for the users which is in section 2.7.

To protect the access of system from unauthorised users, several security checks have been built in. These checks have been discussed in section 2.8.

To keep track of the users of the system a history file is prepared for every transaction takes place. Input output activity of the users would be logged in these files and can
be seen anytime. These history files would be deleted by only privileged users. Details of history files have been dealt in section 2.11. Time out control inside dialogue and determinants of user behaviour have been discussed in next sections.

2.1 WINDOW

A window is a part of the display area (some times the entire display area) used for entering and/or displaying functionally related data. Now a days windows are recommended in advanced user interfaces. Windows allow the user to interact with multiple sources of information at the same time, as if he/she had a set of CRT displays of different sizes at his/her work-station.

The use of windows by a user depends heavily on the tasks the user is trying to accomplish. It is therefore impossible to accomplish an analysis of window and display design without careful consideration of the tasks for which windows are used. Following features of the window have been discussed here.

1. More information

Window techniques may allow relatively rapid access to more information than would be possible with a single frame of the same screen size.
2. Access to multiple sources of information:

A task may require access to independently stored pieces of information. For example, a user may need to refer back and forth to several parts of command he/she is issuing: possible command in command class, command parameters detail, default value of parameters etc.

3. Reminding

Windows can be used to help the user keep track of information likely to be of use in the near future, but that otherwise he would need to expend some sort of effort to remember. For example, menus of commands, a clock window giving the time, a history window containing the several commands most recently issued.

4. External memory

The display of a computer provides the possibility of giving the user an external memory that is an extension of the user's own internal memory. One with which he can remember and keep track of more information than otherwise.

2.2 WINDOW LAYOUT

In our design we have partitioned the screen in four windows. The format is as follows.
| WINDOW 1 | 1 - 2 |
| WINDOW 2 | 3 - 5 |
| WINDOW 3 | 6 - 22 |
| WINDOW 4 | 23 - 24 |

Here window 1 is reserved for permanent display. The items displayed in this window area are:

* Identification of work-station
* Current date and time
* Mail status (only when some mail is there)
* Mode of operation (online/off line)

First two lines of the screen are reserved for this purpose.

Window 2 is for error and brief help. This window is used when the operator is interacting with the system while entering a command, user may commit some mistake. At that time any type of error like user ineligibility, wrong command mnemonic, any parameter related error or any parameter related help are displayed in this window area. Lines 3 to 5 of the screen are used for this window.

Window 3 is reserved for input/output purpose. It may be used for:

* General help
* Various menus and forms
* Results/outputs

This will be the case of general help or results/outputs. This window is mainly used for input/output i.e., for taking
inputs from the operator and displaying outputs. The window 2 and 3 form the dialogue window. Lines 6 to 22 of the screen are used for this window.

Window 4 is for the purpose of emergency. The emergency message would be displayed in this area. Messages are flashed (blinking) and an audio beep is also given with it to attract the attention of the user. User will take immediate corrective step. If corrective action is beyond his approach he/she would intimate immediately to his/her superior. Lines 23-24 of the screen are reserved for this window.

The last three of the windows are used concurrently. But one window may disappear in favour of another. The window size depends on the dialogue element used and may consequently vary during a dialogue session.

2.3 VIDEO ATTRIBUTES

Video attributes are used to emphasize certain important information in order to attract the attention of the user. Video attributes work on the characteristic of the information shown within a entire window, a part of a window, an entire field or within just a part of the field. We have given emphasis on highlighting. Proper care has been taken not to confuse or otherwise overload the user by highlighting. There are a number of areas where
highlighting may be applied, such as:

* Defaults in form
* Optional information entry in form
* Indication of system irregularities and their urgency.

The following highlighting techniques have been considered in our design. They are -

1. Flashing

Information is displayed alternatively as normal characters and as spaces in the prevailing background colour. Since flashing attracts much attention and hence it is restricted for special applications. If some comparatively long text is flashed in emergency, the flashing is slow in order to make the text readable.

2. Underlining

Information is displayed with underlined characters. However, this type of video attribute might make it difficult to observe the cursor on terminals where the underline character is used as the cursor.

3. Inverse video

Information is displayed by inverting the image of the characters, such as going from light characters on a dark background to dark characters on a light background.
2.4 OPTION FOR MENU SELECTION

There may be various ways to go for menu selection. Here we have discussed compatible letter, compatible number, incompatible letter and incompatible number in connection of menu selection. A compatible letter is the first letter of the word it is paired with (A-Assembly; C-Compiler) and a compatible number is the ordinal alphabetical position of the initial letter of the word (e.g. 1 = Assembly; 3 = Compiler; 4 = Delete; 13 = Modify)

Incompatible letter selectors can occur when the letters 'a' through 'z' are used as menu selectors and get paired with words with different initial letters. One result of the menu selection has been shown in figure 2.1.

We summarized the result from this experiment that compatible selectors were superior to incompatible selectors. Compatible letters were the best selectors followed by compatible numbers, while for incompatible selectors, the trend was reversed. Incompatible numbers were superior to incompatible letters.

Norman's theory of action slips predicts that competing activations between the initial letter of a word and an incompatible selector letter should result in more errors than other conditions[26].
MENU SELECTION

COMPATIBILITY

FIG. 2.1
2.5 AN IMPROVED OPTION FOR MENU SELECTION

The mechanism for menu selection discussed in earlier section is not found suitable for our need. Thus neither compatible letter nor compatible number scheme were adopted.

In our proposed command structure and menu selection, we found that more than one command having the same starting alphabet (DISPL_DIR, DISPL_STATUS etc.) within the same command class. Here command class is defined as same action oriented commands are clubbed together to form a class. In other words, command nodes are dependent on command class node. Command class and command code relation has been shown in figure 2.2.

It is clear from figure 2.2 that compile command is a class, where all compilation related commands have been put within this class.

In our considered environment, compatible letter scheme is found quite inconvenient to adopt for menu selection. Because there were more than one option beginning with the same letter.

In compatible number scheme of menu selection, we were limited to options 1 to 26 only. As it is clear A to Z corresponds to 1 to 26 respectively in compatible number scheme. It is found in our case, number of commands within a command class may happen to be more than 26. Our emphasis
FIG. 2.2

- **SYSTEM COMMANDS**
  - **COMPILE COMMAND**
  - **LINK COMMAND**
  - **EDITOR COMMANDS**
    - OPEN A FILE
    - DELETE CONTENTS
    - MODIFY CONTENTS
  - SIMPLE LINK
    - LINK WITH OPTIMIZATION
      - LINK WITH DEBUG
      - LINK WITH NO OPTIMIZATION
  - SIMPLE COMPILATION
    - COMPILATION WITH DEBUG OPTION
      - COMPILATION WITH OPTIMIZATION
      - COMPILATION WITH NO OPTIMIZATION
In designing command name was to provide easiness to the user, which they can remember without much exercise. We have designed our command name completely action oriented. In this way, our command within a class were going beyond 26, whereas compatible number scheme has limitation of 26 commands only. It shows that compatible number scheme for menu selection is not found appropriate.

In compatible letter scheme, we would have taken more than one initial character for selection option. But it was found by observation that if users allowed to type more digit or alphabet, chances of error caused by slip or typing mistake were high. And this would end in a form of action that was not intended by user. It has been established that response time for errors is higher than correct trials. To avoid this situation, we have discarded the scheme of taking more than one alphabet for selection option.

To avoid the above mentioned complications, we have adopted incompatible number scheme for menu selection. The commands, which were used very frequently, were kept in beginning of the sequence. Thus commands having less frequency of use are in the bottom of the sequence.

It has been observed that one or two seconds can be saved every time a menu selection is made. The importance of this result depends on the frequency of menu use. If there are many user using the system and preferring to go for menu
selection, system time as well as user time can be saved to a good extent.

2.6 AN IMPROVED TECHNIQUE FOR MENU NAVIGATION

A menu-driven interface is chosen as the form of human-computer dialogue where menus are arranged in a strictly hierarchical structure. The hierarchical structure is characterised by intermediate levels of breadth and depth of menus as shown in figure 2.3. This takes into account that menus have to be simple enough for easy and quick assimilation.

It has been found that the two most significant obstacles in accessing the information in these systems are cognitive mismatch, caused by the organisation and categorisation of the information, and the difficulties encountered in navigating within the complex menu structures. Deep menu organisation can lead to a loss of orientation of the users within a complex dialogue structure. Related options are semantically categorised and concentrated to facilitate acquisition.

The optimized hierarchical ordering of menus may be appropriate for the design of the human-computer interface in rather small and easy to survey dialogue systems. However in more complex interactive systems, the number of nodes within this hierarchy becomes so large, that the
INDICATES THE CHOICE LEADING TO THE NEXT MENU SEQUENCE

FIG. 2.3
implementation of a conventional menu technique is not effective. The increasing complexity will create major problems, especially for novices and users with minimal training. This is mainly due to the fact that users might feel lost in nontransparent menu driven systems and therefore might take excessively long to respond.

Because menu items have just two possible states, active or inactive, they can be described as bistable. Generally, those choices which have produced the current frame are active (those above the dotted line figure (b)).

The figure 2.3(a) shows a typical hierarchical command structure; (b), (c), (d), and (e) show how the trace of choices and the use of bistable items allow for selective retreat and navigation through the structure.

The choices possible from this frame are normally in active. The two states can be represented on a display screen by differing intensities - bright for active and dim for inactive.

Consider the sample structure described by the directed graph of figure 2.3(a). The menu shown in figure (b) is the one which would be presented after 'b' was chosen from the original menu presented at 'A', and then 'h' chosen from the subsequently presented menu frame. The user can return to the original menu by cancelling the choice from this menu.
This will result in the menu shown in figure 2.3(c). A subsequent choice of 'c' will result in the menu shown in figure 2.3(d). The above mentioned approach will help the user to keep track of the current state (he knows where he is) and how to navigate within the dialogue hierarchy (how to get to another state).

2.7 COMMAND NAME ABBREVIATION

Abbreviation appears in various contexts in computing environments. Systems may require abbreviated forms of command names. Similarly, computer users often generate abbreviations for frequently used commands, command sequences, path routes, file and file directory names, and so forth.

In its most general sense, abbreviation is central to all communication, whether graphemic or iconic, auditory or gestural. The degree of reduction is determined by balancing the demands for efficiency and for intelligibility in communication. Efficiency can be increased through abbreviation, but at the risk of decreased intelligibility. In the computing environment, as well as in the realm of general communication, abbreviated words serve two quite different roles. In one role, the intelligibility is at a premium; in the other, efficiency is stressed.

It is important to note that the names or abbreviations
generated by experts will be different, but not necessarily better than those generated by novices. In particular, the high degree of familiarity with the computing context that leads the user to favour efficiency over intelligibility may be treacherous. Looking over a directory created months or years ago, trying to recall what is in our own cryptically names files, we may be forcibly reminded that what seemed very familiar at the moment was unfamiliar latter.

But by being fully aware of the effects of context familiarity upon our naming behaviour, we may be able to find the right balance between efficiency and intelligibility [66].

It is observed that to encourage the users to abbreviate and thus work faster, the commands could intentially be made long. This approach would be feasible when there are a small number of commands each starting with different letter. Under such conditions the user can abbreviate any command down to one character without causing ambiguity. To make the dialogue understandable to other parties the commands have been made short without compromising their intelligibility. This would help another person who gets a transcript of the dialogue to be able to understand. Benbasat et al[14] have given hypothesis:

- 45 -
1. Command length will have no effect on the amount of abbreviation by subjects.

2. Long command length will lead to more abbreviation compared with short command length.

It has been found that longer the length of the commands the higher is the incentive for the user to abbreviate them for saving input time and effort. This reasoning is based on Zipf's "principle of least effort" which states that length of a word in syllables is inversely related to its frequency of use[22].

It is an established fact that if a user is typing long word, chances of typing error are most probable. In this way abbreviation of command would also reduce the typing error generally caused by users.

2.8 SECURITY CHECK

The term security is used to describe the protection of systems and their contents from destruction. If we wish to secure a system, we will need to achieve a reliable and predictable mode of operation; we also have to provide a protection mechanism to achieve the desired control of access to the data. Following safeguards have been provided to take care of system security issue.
1. Each user is assigned a user name and password. Unless a user is giving his/her correct user name and password, system would not permit to access the system. User name may be known to other users but password is the secret for every user.

2. Each user in our environment would be allowed to issue commands for a few command classes. However, privileged users are allowed for all the command classes.

3. Each terminal attached with DSS is allowed for few command classes to secure the system from unwanted users. However some privileged terminals are allowed for all command classes.

4. User is allowed to change his/her password only. It is advisable to change password periodically. So that no case other user may login in another user's account.

5. If a user after issuing a command realises to abort the command; he can do so. But a user can't abort the command issued by some other user.

2.9 COMMAND FILE EXECUTION

It has been found that if user is allowed to type too many parameters and its value; chances of committing error are
very high. Once error is encountered in the issued command, user could be forced to input again a long list of parameter and values. This situation is found most irritating and uncomfortable for the users.

To overcome this problem, a command file is created. Instead of directly inputting a command, full command details is input in that created file. To execute this command file, a control file is prepared, which will help the system to read the command and its full detail from the previously created file.

Once an error is encountered, proper corrective information would be passed to the user. Now user can easily modify the error causing area of the command file and again he/she can execute the command without much problem.

While issuing command for execution, the user may specify whether to delete the command file after its successful execution is over. Default is not to delete even after the execution is complete.

2.10 FILE AS PARAMETER VALUE

While inputting a command, it is found that some parameters of the command may take long list of values. Probability is very high to type some wrong value in this case. This situation is same as executing a command file. Here instead of taking whole set of command through a file, system would
take value of a parameter of a command from file.

In this case, user is little free, while preparing a parameter-value file. In case of wrong value, he would have to modify error causing area of the file, as he would have received while executing the command.

At the time of validating a command, it is checked, whether value of a parameter may be taken from a file or not. So it is decided well in advance, whether a parameter is allowed to take value from a file or not. Generally parameters, which are expecting to take long list of values are put under this scheme.

While inputting a command, user specifies the file name instead of parameter value. To make it easily understood, user puts a dollar sign ($) before the file name. Dollar sign helps the system to know that values for that particular parameter would be taken from file. System opens that specified file, reads the values and assigns to that parameter.

2.11 LOG FILES

Different type of log files are created for starting from issuing a command to displayng the result. These files are kept for post process analysis. Mainly these files are input/output transaction file and system transaction file.
If something goes wrong in the system, then by the help of input/output transaction file, a superuser (privileged user) can interrogate the matter. It can be traced out the activity of individual users very easily. What all commands are issued by a user, what are the system's response for those issued commands; who is responsible for a particular command etc; these facts help the superuser and administration personnel to take appropriate action against that user if found guilty for the action he performed. These files can not be deleted by general user. Only super-user has the privilege to delete the input/output transaction file.

Contents of these files are user identity, date and time, work-station identity and input/output of the session. System transaction file is created for any type of system response irrespective of the user issued command. Contents of these files are system identity, system response, date and time.

2.12 TIME OUT CONTROL INSIDE DIALOGUE

As user starts session, it becomes necessary to make him/her work attentively and properly for that session. While inputting a command, if user takes long time to give the value of a parameter, system would not allow to work on his/her own will. We have introduced a time limit between two subsequent parameters. If user fails to assign the
value within a specified time limit, timeout would take place and user would be informed about the event. At the end of time out, current command session is terminated and system prompt would be displayed on the terminal. Again he can issue the command for further continuation.

Again if user fails to issue a command within certain time limit, it means terminal is either unattended or user is busy in some other work. In such situation user session is terminated by system automatically. By this algorithm, unnecessary load of the system gets reduced upto a good extent. Now still he is interested to communicate, he would have to start a fresh session.

These two types of time out have been introduced to provide quick response to other users of the system. These time limits have been taken considerably enough by considering human factors in detail. This includes thinking time, inputting time and also pressing return key if required (pressing return key time is applicable in form-filling case).

2.13 DETERMINANTS OF USER BEHAVIOUR

In our work we have tried to focus on the cognitive aspects of the user, which includes learning, performing, and reasoning, and not on the motivational, emotional, or personality aspects.
Users are found engaged in goal-oriented activity. They attempt to accomplish their goal as effortlessly as possible, within the constraints imposed upon them by the structure of the task, by what they know, and by their own information-processing limits[73]. These determinants can be stated in a formula for rational user behaviour:

user's goal +
task structure +
user's knowledge +
user's processing limits => user's behaviour.

To understand what a user does, we must first know the user's goal(i.e., what he is trying to do) because all his actions are organised to accomplish it. We can make no sense of his behaviour without knowing his goal. We must then know the task structure. The most important part of the task structure in human-computer interaction is the computer system's user interface. Next we must know about the user's knowledge, for he can not exploit the structure of the task without effective knowledge of it. Finally, the user has mental processing limits which force him to adopt strategies that keep him within his processing capabilities. Two examples of processing limits are the human's limited short-term memory capacity and and his tendency to make errors occasionally(for variety of reasons).

As we have seen in section 2.5 that list of command class

- 52 -
and command within class, both are large. It has been proved by experiment that efficiency of a user would be high if he/she works for a limited set of task. If same user is assigned the responsibility to work for whole set of task (commands) of the system, efficiency will go down. For a user to get acquaintance for limited set of commands would be easy. He/she can memorize the command, its sequence, procedure etc, without much difficulty. Since he/she has specialisation in particular area, so chances to commit mistake would be very low. On the other end, a user with multiple responsibilities would be baffled and chances of committing error would be high. Thinking time for such users would also be much high.

A case has been analysed for the above purpose. Two groups of users were considered. After recruitment one type of users were given training for limited set of commands and other type got training for multiple responsibilities. Both types of users were working with same system and within the same environment. Their performance was measured time to time. It was found that user with less responsibilities committed less error at any instant of time compared to other type of users. For this purpose, the approximate behaviour of the graph plotted in this connection has been shown in figure 2.4.

While collecting data for right assessment of the two types of users, we have given sincere thought on the dimension of
Here X denotes a user with limited task \((t)\) &

Y denotes a user with multiple responsibilities
\((t+n)\)

WHERE \(n = 1\) to \(N\)

FIG. 2.4
the user behavior. It has been found that user behavior is a complex, having many dimensions, such as:

* Functionality: The range of tasks that the user can do with the system.
* Learning: How long it takes the user to learn how to do a given set of tasks.
* Time: How long it takes the user to do a given set of tasks.
* Errors: How many errors the user makes and how severe they are.
* Quality: How good the output of a given task is.
* Robustness: How well the user adapts to new and unexpected tasks.
* Acceptability: How well the user subjectively rates the system for doing a given set of tasks.

2.14 PROMPTING AND DEFAULTS WITHIN COMMANDS

There remains the issue of what to do when the user, for one reason or another, fails to specify some information that either could or should have been provided. In the narrow sense, this is the situation where a command has been issued, but one or more of its arguments has not been given.
This is, first, the option of disregarding the command entirely - and many systems do so - and, second, if not disregarded, what to do about the missing information.

There are several options for prompting the user for missing information, ranging from a brief listing of the missing argument names to a full display of potential values for each of the missing items along with an easy means of indicating one's choice. For a relatively small set of alternatives, permitting users to choice from among them might be an easier task than asking users to generate the alternative; however, the selection mode becomes much less desirable when the display of alternatives is very complicated or takes a long time to produce.

As an alternative to prompting, it may be possible to assign a value - a default value - automatically to some of the missing arguments. This is one of the most powerful of existing computer system concepts for achieving a user-oriented environment. Essentially the use of defaults constitutes an agreement between user and computer as to what a "normal" or "usual" working environment might be. However, problems can arise e.g., the user does not know of the default assignment system or doesn't understand the default, or the user does not have a convenient means for changing the defaults. Perhaps the computer should (optionally) display assumed defaults to the user. User will get more user friendly behaviour in this case.
In our design, system does not assign the default value automatically, if user fails to give the parameter value within specified time in direct parameter entry mode. But in case of form-filling, system displays the complete form, where arguments having defaults values are displayed with form. These default values are previously chosen value, which the system would automatically accept unless the user alters the value. The default is included in order to reduce routine typing. If user decides to not alter the default value, shown with form, he/she has to press the return key to complete the command arguments.