Chapter Six
Dialogue Design

One of the most critical determinants of user satisfaction and acceptance of a computer system is the extent to which the user feels in control of an interactive session. If users cannot control the direction and pace of the dialogue, they are likely to feel frustrated, intimidated, or threatened by the computer system. Their productivity may suffer, or they may avoid using the system at all.

The extent of system control perceived by the user depends significantly on the extent to which information is provided. This information can make an otherwise difficult and time-consuming task easy and fast. The following guidelines present principles and techniques to make the user's tasks easier.

Status Information
Provide evidence of system status at all times. After an entry, the users need feedback to know whether the system is down, in a loop, in some other error mode, or just in the midst of processing the input. When processing time of a given input is long, feedback on the status of the request is particularly important. *Galitz, 1980*

6.1 Intermediate Feedback
Provide input acknowledgement and progress indicators.

Provide intermediate feedback that acknowledges acceptance of the input and indicates that the system is waiting for processing to be completed. Delays longer than a few seconds can be troublesome for some users unless an indication is given that the system is working. A message such as "Please wait while the system processes your request" may be acceptable. However, provide a dynamic progress indicator whenever it is feasible (Myers, 1985). Dynamic indicators provide feedback on how long the wait may be, give feedback that the processing is continuing smoothly, and hold the user's interest during the wait. *Engel and Granda, 1975; Frey, Sides, Hunt, and Rouse, 1983; Myers, 1985*

Examples:

1 A numeric or graphical percent-done progress indicator provides not only feedback that the request has been accepted, but also an indication of how much waiting remains. For example, a graphical image of a gauge or meter that reads from 0 to 100 percent could display percent-done. On a system without graphical capabilities, numeric percent-done indicators could be used.

2 A counter can provide feedback on how many of a series of operations have been completed. For example, a print routine could show the page number of the page now printing, such as "Now printing page number N", where N represents the current page number.

3 In many situations a count-down counter is preferable to a count-up counter as in the example above. A count-down counter shows the number of operations remaining to be performed, which is often a better indication of the time to completion than the number of operations completed. For example, in a file copying program, the progress indicator could be "N more files to copy".

4 When the number of operations is not excessive, the computer could list each operation as it is performed. For example, in a file copying program, the computer could list each file as it is copied:
"Chapter 1" file successfully copied.
"Chapter 2" file successfully copied.
"Chapter 3" file successfully copied.

6.2 Input acknowledgement
Acknowledge successful completion.

Acknowledge the execution of an input. When processing of a request is completed, some feedback should be given. Often this feedback is implicit in the displaying of the requested data, but when successful execution does not cause any change in the display, a PROCESSING SUCCESSFULLY COMPLETED message should be added. *Engel and Granda, 1975*

6.3 Blank Screen
Do not leave the screen blank.

Always give the user an indication of how to continue. A user left viewing a blank screen has no indication of how to continue, what mode the system is in, or even whether the system is working. *Peterson, 1979.*

6.4 Mode Designator
Display mode indicators.

When the system is operating in a special mode for which a different set of commands or syntax rules are in effect, provide users with an indicator that distinguishes this mode from other modes. Provide differences in headings, formats, or prompts, and use labels to remind the user of the mode that is in effect. For example, the edit mode can be distinguished from the command mode by the presence of a characteristic header or distinctly different screen format and the word EDIT prominently displayed on the screen. *Elam, 1980.*

6.5 Transaction Type
Display the current transaction type.

Display the transaction category name of the current transaction. Within the same mode there may be categories of transactions that serve very different kinds of functions. For example, a data base application may have separate transaction categories for reports, adding data, changing data, deleting data, menu screens to select other transactions, and HELP screens for online assistance. Displaying the category name (REPORT, ADD, CHANGE, DELETE, MENU, or HELP) prominently on the screen helps the user to maintain a frame of reference.

6.6 System Entry
Explain down time.

If a user will not be able to get onto the system, send a message telling why and approximately when the problem is expected to be corrected. For example, display SYSTEM DOWN FOR MAINTENANCE UNTIL 9:30 AM. Avoid "as soon as possible" messages; make an estimate and update it as required. *Brown et al., 1980, 1983*

6.7 Explicit Options
Display available options.

Rather than requiring users to remember which options are available at each step in the dialogue, display a list of the currently active options. Temporarily inactive options (those that are not applicable in the current stage of the dialogue) may also be displayed if their inactive status is shown. *Brown et al., 1980,1983; Elam, 1980; Peterson, 1979*

6.8 Memory Requirements
Display needed information.

Do not require the user to remember information not displayed on the current screen. The user must not have to decide what action to take from memory. inaccuracies of memory will lead to errors and lost time. *Brown et al., 1980, 1983; Engel and Granda, 1975*

6.9 Sequence Continuation
Displays indicate how to continue.

Indicate on each screen the user response that is necessary to continue the interaction sequence. Do not leave the user viewing a screen with no indication of how to continue.

USE

Data Base Status Data is current through March 1985 To continue, Enter NEXT ID or press PF 11.

DON'T USE

Data Base Status Data is current through March 1985.

Brown et al., 1980, 1983; Engel and Granda, 1975

6.10 Multi-screen Transactions
Place frequently used steps first.

For transactions requiring multiple screens, locate the most frequently requested data on the earliest screens. Allow users to skip later screens in the sequence if this is reasonable in the transaction. *Galitz, 1981*

6.11 Visible Effects

What you see is what you get.

The results of user entries and actions should be directly and accurately portrayed on the display.

Example:

A word processing program should enable the user to view on the screen exactly what the pages will look like when printed. Users find it very difficult to ensure that proper spacing, margins, page breaks, and alignment have been achieved unless they can view an exact rendition of the printed page before printing. If this is not possible in the input mode, the program should provide a "FORMAT CHECK" mode. This mode incorporates the requested margins and tabs and shows what the printout will look like with all non–printing characters and embedded commands removed. *Smith et. al., 1982.*

6.12 Access to Settings

What you see is what you've got.

Provide ways for the user to easily determine the settings of all dialogue parameters. For example, in a word processing application, the user should be able to quickly determine the settings of parameters such as line spacing, character size and font, margins, and tabs. *Shneiderman, 1986*

6.13 Reversible Actions

Provide an "UNDO" function.

Provide the capability to recover from unwanted or incorrect actions. An "UNDO" function, which reverses the effect of the last action taken, is often used to provide this capability. The ability to immediately recover from an inadvertent or erroneous action (or an action with different results than expected) has several advantages for users:

1 It can eliminate the necessity to reinitiate a whole series of transactions that led to the previous step in the dialogue.

2 One can recover from mistakes that might otherwise have led to the destruction of data, requiring reentry or perhaps permanent loss of those data .

3 It can minimize users' fear of getting into a situation from which they do not know how to return.

4 By trial–and–error exploration, users can learn the system or expand their knowledge to functions they have not previously used .

Example:

The user is not sure which of the titles shown on a selection list performs the desired function. He chooses the one that seems most likely, but the result is not what was he wanted, and his original data have been modified. The UNDO function permits him to reverse the last transaction and thus avoid having to reenter or reconstruct his original data.

Note:

The need for reversible actions applies at many levels, from the ability to back up with cursor control or tabbing keys to the need in some applications to retrieve the previous version of a file. Actual inverse functions (such as "tab left" and "tab right") tend to be more natural and more easily understood than UNDO (for those functions for which an inverse exists). *Foley and Wallace, 1974; Galitz, 1980; Shneiderman, 1982b; Smith et al., 1982*

Menus

A menu is a list of options from which a selection or selections can be made by the user. It provides to the user an explicit list of available options, permitting selection by recognition rather than requiring recall. Menus provide a useful technique to make computer systems more accessible to novice users, less demanding on the memory of infrequent users, and , if properly designed, more efficient even for experienced users. If not carefully designed to accommodate both experienced and novice users, however, menu dialogues can be intolerably slow and frustrating for expert users. *Perlman, 1984*

Modern dialogue designs have expanded the definition, responsiveness, and functionality of menus considerably over traditional menu techniques. Traditionally, menus were full-screen text lists of options. The user was required to work step by step through a hierarchical series of menu screens with (typically) a long response time between each selection. Advancements in display technology, computer memory, local intelligence, graphics, and user interface innovations have permitted menus to be much more flexible and efficient. Menu windows, pop-up and pull-down menus, graphical and iconic menus, and nearly

instantaneous menu response time have become commonplace. Many of the traditional objections to menu systems for experienced users need not apply.

6.14 Main Menu

Make the menu easily accessible.

When using a menu system, provide rapid access to the main menu at all times. The user should not have to backtrack to return to the starting level in a hierarchical menu system. This capability can be provided by dedicating a program function key, a light pen or touch field, or a cursor entry field to display the main menu.
Brown et al., 1980, 1983

6.15 Multiple Paths

Provide menu bypass capability.

Permit the user to request displays or options from non–menu screens and from menus not showing the desired selection. Menus are helpful for novices to the system or to a part of the system, but working step–by–step through several levels of a menu hierarchy is slow and unnecessary for experienced users. *Galitz, 1981*

Examples: Direct routing techniques include:

1 command language entries,

2 macros (sequences of commands or actions initiated by a single input),

3 "stacking" menu requests, and

4 typing the memorized number, name, or ID or the desired display.

6.16 Entry Stacking

Allow type-ahead entry stacking.

permitin

When possible, permit stacking of inputs or multiple entries. When experienced users are responding to the first of a sequence of selection screens and know before seeing them how they will respond to the next

several screens, they should be able to input several entries at the same time. Sequential entries can be separated by a special character; a semicolon is recommended. For example,

COST CATEGORIES.

CODE	CATEGORY
L	Labor
M	Material
Enter selected code: L R T	

(The R and T entry codes are normally displayed as selections at the next two menu levels, but three entries can be made simultaneously by users who have memorized the desired codes. The two intermediate menus are thus bypassed.) *Engel and Granda, 1975; Shneiderman, 1983a*

6.17 Menu Wording

Use clear wording of menu selections.

Spell out the selection options on a menu list. Choose words that make the function of each option clear and distinct from the other options. Minimize the use of abbreviations in menu items.

6.18 Consistent Titles

Make screen titles consistent with menu wording.

Use the same title for an option in a menu listing as the screen title of that selection. Users will expect to see the same words that they selected from the menu when the selected screen appears. A different screen title may lead them to believe that they made a selection error.

6.19 Menu Order

List menu items in appropriate order.

Menu items can be listed by several different ordering schemes. The appropriate ordering scheme depends on the nature of the items and their typical uses, but a systematic ordering technique should be applied. Possible orders include the following:

1 chronological
(time ordered),

2 numerical
(listed by size or quantity),

3 alphabetical
(alphabetical order of item names),

4 sequential
(listed by typical sequence of use),

5 functional
(functionally related items listed together),

6 importance
(most critical items listed first), and

7 frequency
(most often used items listed first).

Brown et al., 1980,1983

6.20 Inactive Menu Options

Display only active menu options.

When designing selection menus, do not include options that are planned, but not yet implemented. Unimplemented options clutter the screen and may send the user down dead-end transaction paths. *Smith and Aucella, 1983.*

Example:

When the "Inventory Data Base" is planned for future implementation, but is not online, don't display its name on the main menu.

USE

```
ID = MM                        MAIN MENU                    Page 1 of 1

Enter the selected Data Base ID as the NEXT ID.

ID    DATA BASE NAME

M     Manufacturing Data Base
A     Accounting Data Base
S     Scheduling Data Base

NEXT ID: ___
```

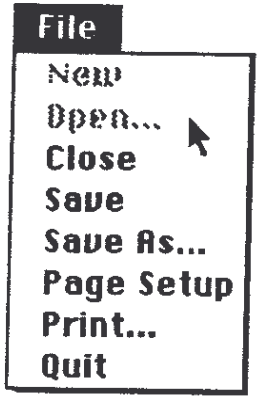

DON'T USE

```
ID =MM                         MAIN MENU                    Page 1 of 1
Enter the selected Data Base ID as the NEXT ID.

ID    DATA BASE NAME

M  Manufacturing Data Base
A  Accounting Data Base
S  Scheduling Data Base
I  Inventory Data Base

NEXT ID: ___
```

Exceptions:

Some systems display options that are momentarily inappropriate at the current stage of the dialogue in shaded characters while displaying currently active options in normal characters. This technique is valuable in providing the user with a consistent, familiar menu selection list, yet designating when some of the options are momentarily unavailable or inapplicable in the current situation.

Example:
The figure below is a menu of File options. "New" and "Open..." are implemented options, but are not appropriate in the current situation.



Soft Machine Controls

Nakatani and Rohrlich (1983) have described a "soft machine" philosophy for designing user dialogues with computer systems. A soft machine is a software-driven emulation of a hardware display/control device, such as a panel of pushbuttons, indicator lights, knobs, dials, or meters. The "hard machine" display is emulated by a CRT or plasma panel facsimile and the control activation is emulated by a cursor control selection device or by a touch sensitive overlay.

Hard machines (conventional stoves, toasters, stereos) have some ease-of-use advantages over traditional computer interfaces such as command languages. They are simpler and more intuitive to use. Their functions are usually mapped one-on-one with their controls. Their appearance thus gives an indication of how they are operated. However, they also tend to be inflexible, difficult or impossible to reconfigure to support new functions, and limited in the number of functions they can support in comparison with the open functionality of computers.
By emulating hard machines in a software-controlled, reconfigurable medium, soft machines can capture the benefits of both hard machines (for ease and naturalness of use) and computer systems (for flexibility, functionality, and ease of reconfiguration). A soft machine interface can

emulate a large number of different hard machines on the same physical device, and can be reconfigured in seconds to replace a module that is not currently required. *Nakatani and Rohrlich, 1983; Seminara and Eckert, 1980*

Tognazzini (1985) has described several kinds of soft controls with recommendations for their use: buttons, check boxes, radio buttons, and dials. All of these techniques, which will be described in the following sections, use a pointing and selecting device to perform direct, cause-and-effect manipulation on graphical objects and their labels.

6.21 Soft Machine Menus

Consider presenting menus as soft controls.

Menus are often a good application of the soft machine philosophy. Options can be presented as graphical facsimiles of labelled pushbuttons or checklists. A cursor pointing and selection device, such as a mouse or light pen, or a touch sensitive overlay is used to "push" the selected button.

6.22 Buttons

Use soft buttons to initiate actions.

Soft buttons are small displayed objects, such as boxes or facsimiles of pushbuttons, labelled with text. Pointing and clicking on the button (with a mouse) or touching the button (with a finger on a touchscreen) initiates the action indicated by the button's label instantly after pressing. Buttons are typically used to permit the user to select among a small number of options (typically about two to five) to be activated immediately after the button press. Example of functions for which buttons are commonly used include acknowledging messages and cancelling or proceeding with actions which require confirmation (such as deleting a file). *Tognazzini, 1985*

Example:
The "Done" and "Cancel" buttons in Figure 6.1 are examples of this type of soft control. They are used by a document processing program to close the Table Properties window shown in the figure. "Done" causes the new configuration to take effect, while "Cancel" returns to the previous configuration.

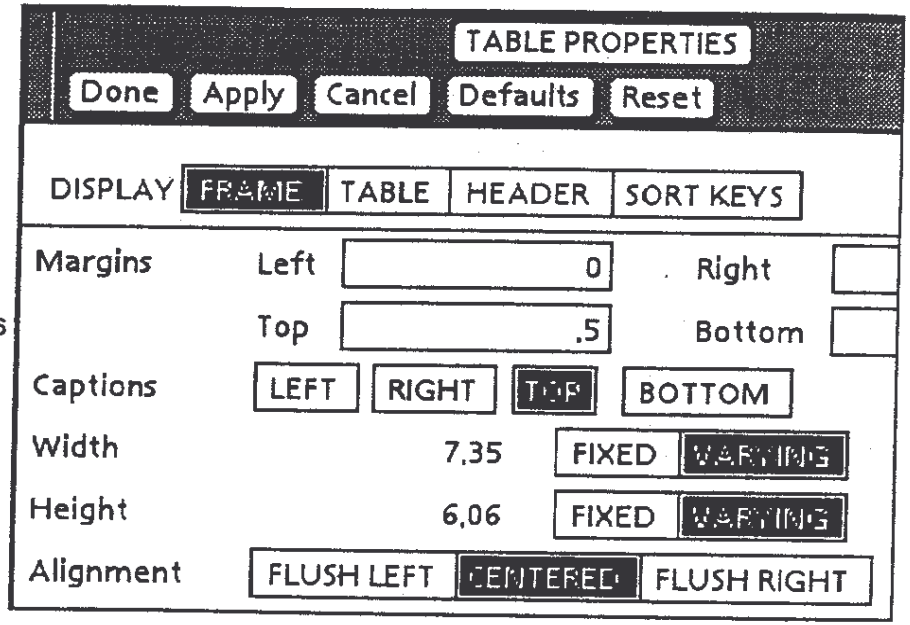


Figure 6.1
"Table Properties" Window - Example of Various Soft Machine Controls.

6.23 State Boxes
Use state boxes to turn options on and off.

A state box, also called a check box, is a labelled box that controls the ON or OFF state of a parameter. By touching or clicking the state box with a cursor control device, the user can change the state of that parameter.State boxes show the current state of each parameter, typically by highlighting those that are ON or placing a mark beside them.

A series of state boxes can be presented in the same "menu", permitting the user to configure the whole cluster of related parameters at the same time. Unlike buttons, state boxes do not have an immediate effect. Instead, when the user signals that configuration is complete (often by a "Done" or "Enter" soft button), the reconfigured set of parameters take effect. *Tognazzini, 1985*

Example:
The items in Figure 6.1 under "Captions" are state boxes. The user can turn any combination of these caption location options ON or OFF by simply clicking on any box to change its state. Options that are ON are displayed in highlighted boxes; those that are OFF are displayed in unhighlighted boxes.

6.24 Choice Boxes
Use choice boxes for mutually exclusive choices.

Choice boxes (also called radio buttons after the station pre-set buttons on a car radio) are used to select among two or more mutually exclusive options. When one of a set of options is selected, the previously active option is automatically de-selected (like the car radio station pre-set buttons). Choice options are displayed as connected boxes to differentiate them from state options, which are displayed as unconnected boxes. *Tognazzini, 1985*

Example:
In Figure 6.1 the options under "Alignment" are choice boxes. The highlighted "CENTERED" option indicates that "CENTERED" is the currently selected option. Only one of these three alignments (FLUSH LEFT, CENTERED,or FLUSH RIGHT) can be in effect for a given table. Thus, if "FLUSH RIGHT" were selected, "CENTERED" would automatically be de-selected.

6.25 Soft Dials
Use soft dials for magnitude or position controls.

Soft machine techniques can also be used to emulate quantitative physical control/display devices, such as dials and meters. For example, some computer systems use a "scroll bar" to provide a technique for displaying and controlling the vertical or horizontal position in a file or on a virtual page. *Tognazzini, 1985*

Example:
Figure 6.2 shows several controls which are operated by moving an indicator directly using a cursor control device.

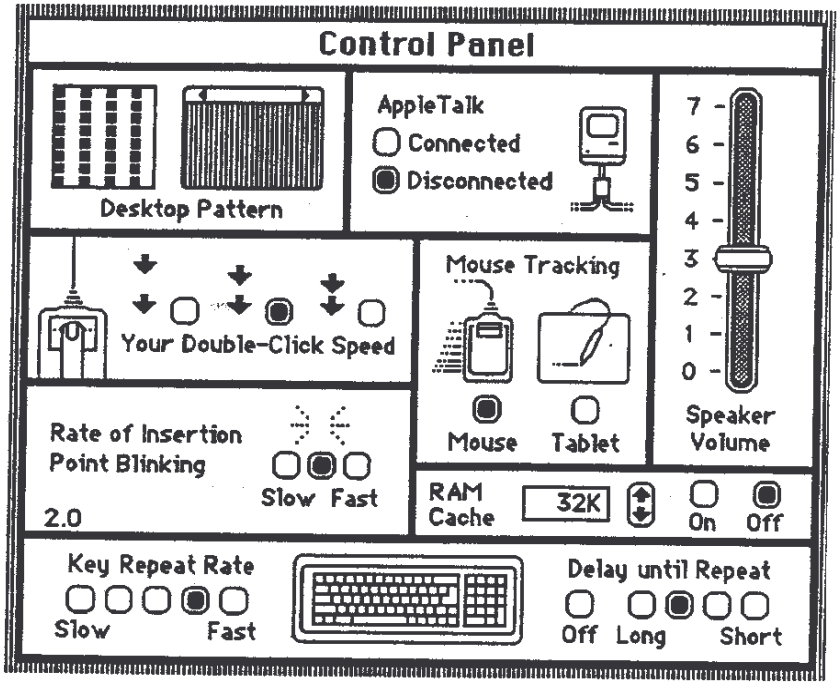


Figure 6.2.
Macintosh "Control Panel" Example of Various Soft Dials

6.26 Combined Controls
Use combinations of soft controls together.

In many human-computer interface design situations, soft control techniques such as those described above can be combined in the same window or transaction. This allows greater flexibility and permits setting or changing the configuration of parameters more quickly and easily, and in fewer transactions, than is typically possible using traditional menu systems.

Example:
Figure 6.1 shows a combination of buttons, state boxes, choice boxes, and dials used to completely configure an array of character format options in the same window with one transaction.

6.27 Soft Control Conventions
Conventions differentiate among soft control types.

Use different display conventions to discriminate among different types of soft controls. These will cue the user on whether a given option is an immediate-execute (button), a delayed-execute with multiple independent choices (state boxes), or a delayed-execute with mutually exclusive choices (choice boxes). A display convention for each should be established and observed consistently throughout the system.

Example:
In Figure 6.1 immediate action buttons are round-cornered rectangles inside the header. The choice boxes are connected boxes that use highlighting to indicate which mutually exclusive option is in effect. The state boxes are separated boxes that use highlighting to indicate "ON".

Commands
Command languages permit the user to control the computer dialogue by entering instructions to the computer, traditionally as type-in words, codes, or abbreviations. Modern dialogues often present command options on the display and permit the user to enter a command by pointing to its displayed label. These techniques have made the distinction between a command language and a menu system somewhat blurry. This section will focus primarily on dialogues that require the user to type memorized words, abbreviations, or codes, although alternatives will be mentioned.

A command language is often distinguished from a programming language by its immediately responsive, interactive nature. The programmer composes a long list of computer instructions for later compilation and execution. The command language user's instructions are carried out individually as the user enters them.

Command dialogues usually require extensive memorization of command names, arguments, and syntax. This can create problems for novices and intermittent users, but may be managed easily by expert frequent users. For experts, commands can offer short cuts, access to deeper levels of the system's functionality, and custom-tailoring of the dialogue. The challenge for human-computer interface designers is to identify and incorporate command naming, syntax, selection, and entry strategies that make the system easier to learn for novices and easier to use for both novices and experts.

6.28 Misspelling

Recognize common misspellings of commands.

When possible, a system should recognize common misspellings of a command and display a message showing the user the correct command. By confirming this message, the user can execute the command. If the user does not want to execute the system's corrected command, he or she can cancel rather than confirm. *Hayes, Ball, and Reddy,1981*

6.29 Similar Commands

Misspelling does not cause unintended actions.

Though the system should be tolerant of common misspellings, spelling errors should not produce valid system commands or initiate processing sequences different from those intended. Consider possible confusions with existing commands when selecting new command words. *Brown et al., 1983; Galitz, 1981; Shneiderman, 1979*

Example:

TO MEAN	USE	DON'T USE
Define a new procedure	PROCEDURE	PROCEDE
Calculate the difference (gross minus costs)	PROFIT	PROCEED
Show insurance policy value	INS	INS
Ensure that forms are completed	CHECK	ENS

6.30 Truncated Commands

Allow truncation of commands.

Permit abbreviation of inputted commands. The system should recognize a command from the first one, two, or three letters (as necessary to make it distinct from other commands). Novice users can type in the entire command, while experienced users can abbreviate or truncate it. The system should accept either form. For example, if a P uniquely identifies a print command (no other commands start with P), it should be the user's option to enter either PRINT, PR, P, or any other truncation to initiate a print command. *Ehrenreich 1981, 1985*

6.31 Abbreviation Rules

Use consistent abbreviation rules for commands.

Follow a rule or set of rules for selecting command abbreviations. Truncation rules have been recommended by Ehrenreich (1981), but Grudin and Barnard (1984, 1985) suggest that the best abbreviation strategy may depend on whether the user is trying to construct the correct abbreviation from the word (encoding) or trying to reconstruct a word from the abbreviation (decoding) . Encoding is required to use abbreviations as type-in commands. Decoding is required to interpret abbreviated labels or data on a display.

In a subsequent survey of abbreviation experiments, Ehrenreich (1985) concluded that encoding an abbreviation is easier when a simple abbreviation rule is followed and taught to the user. Truncation has consistently proven to be the abbreviation rule of choice for encoding (remembering the abbreviation given the word). Decoding (remembering what a given abbreviation stands for), however, has not shown a consistent advantage for truncation rules (Rogers and Moeller, 1984). For example, "ACCO" may be the best type-in command abbreviation for "account" if truncated abbreviations are used for all commands. However, "ACCT" may be a better displayed label abbreviation to identify a data field.

6.32 Irreversible Commands
Require confirmation of destructive commands.

Require the user to confirm before executing a command that would result in extensive, irreversible changes. When feasible, the message requesting confirmation should provide a specific advisory about the irreversible consequences. For example, if the user enters a LOGOFF command before saving the data file he or she has created, a message such as "Your data will be lost if you logoff now. Do you want to save the file before logoff? (Y=yes, N=no, C=cancel command)".

6.33 Command Location
Use a consistent screen location for command entry.

When using commands in a formatted screen or window display, provide a command line(s) or field(s) in a consistent location on the screen or window. Granda et al. (1982) found that locating the command line near the bottom of the screen resulted in shorter command entry times than when the command line was located at the top of the screen. The number and magnitude of head movements was also less when commands were entered at the bottom, suggesting the potential for reducing the muscular fatigue sometimes associated with prolonged display terminal use. *Granda, Teitelbaum, and Dunlap, 1982; MIL-STD-1472C, 1981*

6.34 Command Consistency
Use consistent command syntax, arguments, and grammar.

Develop and follow consistent conventions for:

1 command syntax (the rules by which commands and their arguments are entered),

2 arguments (command extensions or qualifiers), and

3 grammar (the noun/verb, singular/plural, conventions used in commands).

Consistency reduces the memory load and uncertainty in recalling commands by presenting a familiar, predictable pattern. *Shneiderman, 1986*

Example:
Use a consistent syntax in the way that commands and their objects are connected.

USE	DON'T USE
COPY 5 TO 15 DELETE 5 TO 15	COPY 5 TO 15 DELETE 5:15

6.35 Command Syntax
Use command first, object second syntax.

For commands that specify an action (verb) and an object on which that action is to be performed, the syntax should be verb first, object second.

USE	DON'T USE
INSERT TEXT MOVE CURSOR	INSERT TEXT CURSOR MOVE

Exception:
In point-to-select dialogues, it may be preferable to select the object first, then the action to be performed on that object. *Shneiderman, 1986*

6.36 Hierarchical Commands
Use hierarchical command structures where applicable.

For systems with a large number of operations that are hierarchically related, or that can be specified by combining a selection from each of several independent categories, command hierarchies can minimize the number of command words to be memorized. For example, an action-object-destination hierarchy permits the user to specify a massive number of possible outcomes using only a few action names, a few object names, and a few destination names.

6.37 Congruent Commands
Use congruent command pairs.

For operations that have inverses or other counterparts, such as opening or closing a file, requesting the previous page or the next page, or inserting and deleting data, use congruent pairs of commands. Congruent pairs are usually formed from opposites or inverses.

USE	DON'T USE
READ/WRITE	READ/CLOSE
OPEN/CLOSE	OPEN/SAVE
UP/DOWN	PREVIOUS/DOWN
YES/NO	YES/OFF

6.38 Distinctive Commands
Choose distinctive, specific words for commands.

Commands that are distinctive, infrequently encountered, and specific words are more easily remembered than general and frequently used words. The word chosen for a command operation should discriminate that operation sharply from alternative interpretations or related operations. Less common words are also easier to associate with an operation because there are usually fewer competing associations for uncommon words.

USE	DON'T USE
INSERT	ADD
DELETE	REMOVE

Barnard, Hammond, MacLean, and Morton, 1982; Black and Moran, 1982; MIL-STD-1472C, 1981

6.39 Command Menus
Consider using command menus.

Pop-up or pull-down command menus offer an alternative to typed-in commands that makes the options explicit and requires no typing. These menus can be continually accessible, yet take up minimal screen space when not in use. Providing both command menus and type-in command counterparts supports novices, intermittent users, and experts. This multiple path approach also encourages the graceful evolution of a user from novice to expert. *MIL-STD-1472C, 1981*

6.40 Command Macros
Permit user- defined command sequences.

Macros are user-defined sequences of several commands that can be named, saved, and subsequently executed as a higher order command by simply entering the macro name. This capability is a valuable short-cut for experienced users and those who perform routine or repetitive operations.

6.41 Command Help
Provide access to command prompts.

Provide for prompts or HELP routines to show lists and definitions of valid commands, their valid arguments, and their syntax. *MIL-STD-1472C, 1981*

6.42 Command Punctuation
Minimize special characters in commands.

Use a minimum of punctuation and special characters in commands, especially those used in unfamiliar ways (other than standard punctuation). Keywords are often easier to understand and remember than special characters. If single letter abbreviations of the keywords are permitted, there may be no more key strokes required by the keyword approach than by special characters. *MIL-STD-1472C, 1981; Shneiderman, 1986*

System Response Time
System response time has been the object of considerable concern and attention, numerous studies and experiments, and varying recommendations for optimal response times. Adequate system response time is critical to user performance, accuracy, productivity, and satisfaction with the computer system. Response time requirements can also be major drivers of the cost of a computer system, so tradeoffs are usually required to develop a cost effective system.

For all its importance and the attention it has received, response time has proven to be one of the toughest categories for developing defensible guidelines. In fact the most common recommendations for optimal system response times today are derived from Miller's (1968) introspective analysis. Miller developed his recommendations by extrapolating from the maximum delays people find acceptable in conversations with other people.

Because the "optimal" response time may vary with the user's task, the dialogue used, the skill of the user, the user's expectations about what response time should be, and a host of other factors, simple universal answers to the question, "What should the maximum system response time be?" have not been forthcoming. The guidelines to follow, especially those that mention specific times, should be considered as general guidance to be qualified by task-specific, system-specific, and user-specific considerations.

A key factor determining acceptable response delays is level of closure. Delays that occur after the user has completed a planned sequence of actions are less disruptive than those that occur in the middle of a sequence. A delay after completing a major unit of work may not bother a user nor adversely affect performance. Delays between minor steps in a larger unit of work, however, may cause the user to forget the next planned steps. In general, actions with high levels of closure, such as saving a completed document to a file, are less sensitive to response time delays. Actions at the lowest levels of closure, such as typing a character and seeing it echoed on the display, are most sensitive to response time delays.

User response time, often called think time, is the interval between the presentation of information on a display and the user's next response. System response time affects user response time in several ways:

1 Delays of longer than a few seconds can result in the disruption of thought and short term memory for the next planned action(s).

2 When users expect long delays, they will take more time to organize and check their responses before entry, because the time cost to correct an error is high.

3 Conversely, when response time is very fast, users will respond quickly, not afraid to risk an error because an error can be quickly corrected.

4 System delays can reduce user response time under some circumstances, if the users plan and organize their next response during the waiting time.

Table 6.1 summarizes the response time criteria from MIL-STD-1472C (1981). These criteria are shown here with the same caveat MIL-STD-1472C gives:

System response time is highly application dependent. Therefore, the response times stated in this section are intended to be guidelines. Real-time systems (e.g., fire control systems, command and control systems) may require system response times more stringent than the guidelines, whereas, non-real-time systems may be designed with relaxed response times. MIL-STD-1472C, 1981

Table 6.1
Response Time Recommendations of MIL-STD-1472C

Action	Definition	Max. Response Time (Seconds)
Key Print	Press key until appearance of character	0.2
XY EntEntry	Select field until visual verification	0.2
Point	Input of point to display of point	0.2
Sketch	Input to display of line	0.2
Local Update	Change using local data base	0.5
Page Turn	Request until first few lines visible	1.0
Function Selection	Selection until respose	2.0
Host Update	Change using readily accessible host date	2.0
Simple Inquiry	Display of commonly used message	2.0
Error Feedback	Entry of input until error message appears	2.0
File Update	Update requires access to host file	10.0
Complex Inquiry	Seldom used calculations in graphic form	10.0

6.43 Dialogue Considerations

Consider response time in choosing a dialogue.

When developing a human-computer interface for a system in which you have no control over the response time, take the system's response time characteristics into account when selecting the dialogue approach. Menu dialogues, for example, make quick responses critical to efficient user interaction. Immediate, directly observable results are also crucial elements of a direct manipulation dialogue. With a full-screen transaction dialogue, such as form-filling, short response times are also desirable, but moderate response times are less disruptive for form-filling dialogues than for menus and direct manipulation. *Martin, 1973*

6.44 Echo Delay

Echo user entries instantaneously.

Response time for echoing user entries, such as the time between pressing a character key on the keyboard and showing that character on the display, should appear to the user to be instantaneous. This requires delays of no more than 0.1 to 0.2 seconds. In point-to-select dialogues, the highlighting or marking of a selection is considered an echo, and should appear instantaneously. Auditory signals associated with selection should also occur instantaneously. *Gallaway, 1981; MIL-STD-1472C*

6.45 Interactive Delay

Respond to interactive requests in two seconds.

For an effective interactive dialogue, response time should be no more than 2 seconds, and preferably less.

IBM reports by Doherty and Kelisky (1979) and Thadhani (1981) have shown huge increases in the number of transactions programmers enter per unit time when response time is reduced to less than one second. They have characterized this as a productivity increase, but transaction volume is not necessarily a measure of usable work productivity. In fact, users tend to work faster and less carefully, make more errors, and generate more unproductive transactions when response time is especially fast, probably because the time cost of errors is less. *Gallaway, 1981; Shneiderman, 1980, 1983d*

6.46 File Delay

Load or save files within ten seconds.

Delays of up to 10 seconds may be acceptable for file access or updates, because these requests often come at major closure points in the user's action plan, and because users often expect and accept delays during file updates. Where practical, however, shorter response times are desirable. *MIL-STD-1472C, 1981*