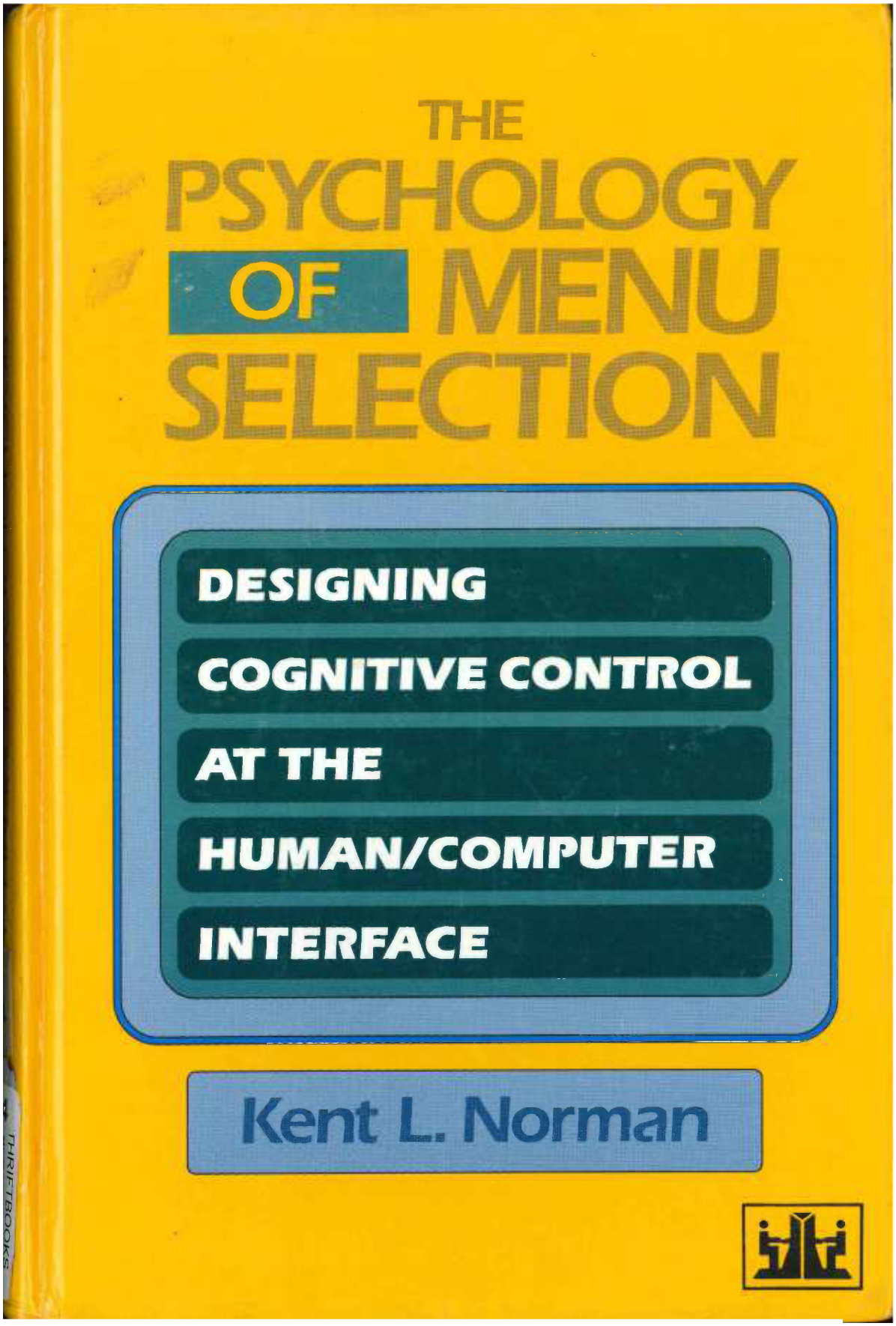
THE
PSYCHOLOGY
OF MENU
SELECTION
DESIGNING
COGNITIVE CONTROL
AT THE
HUMAN/COMPUTER
INTERFACE
Kent L. Norman
THRIFTBOOKS

NORMAN
THE PSYCHOLOGY OF MENU SE
DESIGNING COGNITIVE CONTROL AT THE HUMAN/COMPU
SCI
ENGR.
754
THRIFTBOOKS
12 03181 754

The Psychology of Menu Selection:

Designing Cognitive Control of the Human/Computer Interface

Kent L. Norman

Department of Psychology
University of Maryland

ABLEX PUBLISHING CORPORATION
NORWOOD, NEW JERSEY

Cover designed by Thomas Phon Graphics

Printed in the United States of America

Library of Congress Cataloging-in-Publication Data

Norman, Kent L.
The psychology of menu selection : designing cognitive control at the human/computer interface / Kent L. Norman.
p. cm.—(Human/computer interaction)
Includes bibliographical references (p.) and index.
ISBN 0-89391-553-X
1. Human-computer interaction. 2. System design—Psychological aspects. I. Title. II. Series: Human/computer interaction (Norwood, N.J.)
QA76.9.H85N67 1990
005.1—dc20 90-1198
CIP

Ablex Publishing Corporation
355 Chestnut Street
Norwood, New Jersey 07648

3

Tasks and Flow of Control

Menu selection systems are used to perform a variety of types of work from word processing to information retrieval, from computer-aided instruction to computer-aided design, and from teleconferencing to file management. In each case, a task analysis would indicate that work is performed by executing a number of component tasks and subtasks. Furthermore, such analyses would reveal a plan of attack that specifies the order in which the tasks and subtasks must be executed. Experienced workers have a fairly clear idea about what the component tasks are and have learned a number of protocols, rules, and strategies for applying the component tasks. When computers are used to automate part of the task, there should be a close match between user-defined component tasks and system-defined functions. Few things can be more frustrating than wanting to perform some function that seems necessary and logical to the user but that is not available on the system. Furthermore, there must be an agreement between the order in which the user would like to perform the tasks and the order allowed by the system. Users should be allowed to perform tasks in an order that makes sense to them, even though it may not be optimal in terms of computer processing. The overriding principle is that computers should be designed to increase human control rather than optimizing computer processing at the expense of human control.

This chapter extends the theory of cognitive control and discusses general types of tasks accomplished by menu selection. These tasks include the specific functions of selecting items frame by frame and the global task of traversing a menu structure for specific applications. The theory developed and the distinctions drawn in this chapter serve to set the foundation for understanding the empirical research conducted on menu selection.

3.1 TAXONOMIES OF TASKS AND INFORMATION STRUCTURES

In designing a menu selection system, it is necessary to develop a taxonomy of the tasks and subtasks that the user desires in performing the work. Tasks need to be considered at the cognitive level; that is, they

need to be defined in terms of the components that the user thinks about when he or she is formulating a plan of action. They must not be too elemental nor too global in nature. Instead, they must be gauged to the level of action as defined by the user. If actions are too elemental, the menu structure will be more complex than it needs to be. If actions are too global, the menu structure will be simple but it will overshoot the user's goal.

For any task, the designer must create a list of all the actions implemented by the system. In formal programming languages, this list constitutes the set of allowable program statements and functions. In menu selection systems, this list becomes the set of menu items that evoke actions or changes. An analysis of the task is necessary for the designer to determine what functions need to be implemented. In developing a taxonomy, one may take either a *top-down* or a *bottom-up* approach (Chin, 1986). In a top-down approach, the designer lists major top-level functions which are then refined in greater and greater detail. In a bottom-up approach, the whole list of specific functions is analyzed and organized into groups. Which approach is used depends on the particular task and functions. For example, in text editing one may start with a top-down approach as shown in Figure 3.1. The major task components are (a) file functions, (b) printing, (c) text modification, (d) formatting, and (e) browsing. Each of these may then be refined to more specific functions. The top-down approach has the advantage in that as functions are added they are incorporated in a hierarchical structure.

On the other hand, when a list of specific functions already exists, the bottom-up approach may be used. Tasks that have been manually performed in the past may be analyzed into subtasks. The subtasks are then clustered into groups. Figure 3.2 gives an illustration of a task analysis of a message handling system.

3.2 HUMAN VS. COMPUTER CONTROL OF FLOW

Once the list of tasks has been determined, designers must consider the order in which tasks are performed. The order may be either user-directed or computer-directed. In user-directed tasks, the user starts from a plan of action and directs the computer to perform those tasks in that order. Although command language has traditionally been the mode of interaction for user-directed tasks, menu selection is playing an increasingly important role. The two major problems with command language are that the user must remember the command (both its semantic function and syntactic structure) and correctly enter the command via the keyboard. Menu selection can often be used more effec-

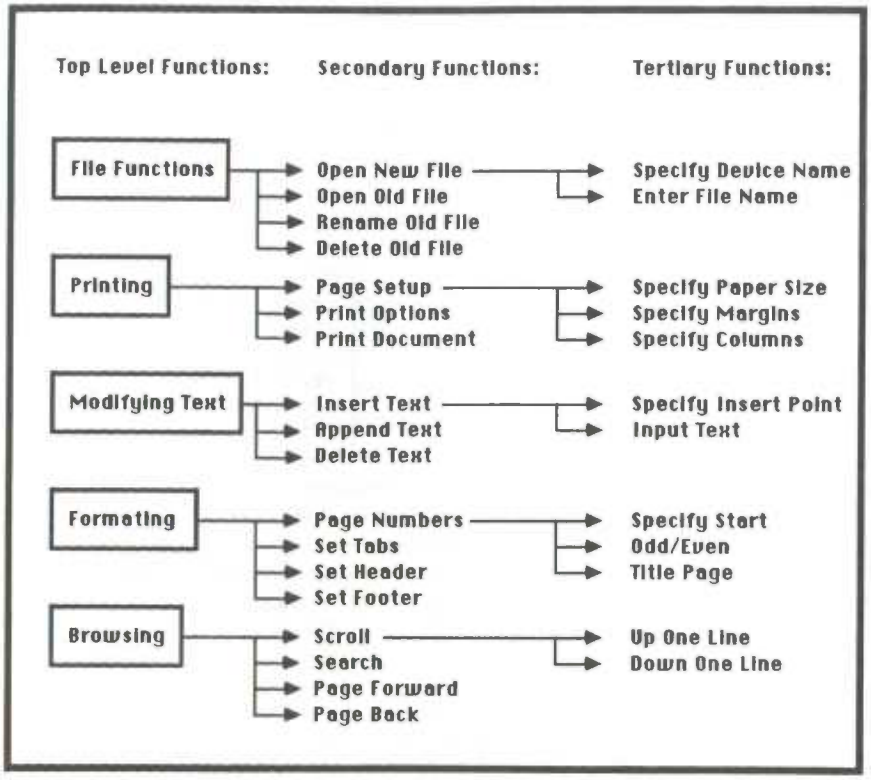

Figure 3.1. Example of a top-down analysis of functions in a text editor.

tively without burdening the user with the added memory and time required to type commands.

In computer-directed tasks, the designer formulates the plan of action and software elicits input from the user in a prescribed order. Computer-directed tasks lead the user through a series of queries in structured tasks that must be carried out in a specific order. Prompts and form fill-in screens are often used to guide users in computer-directed tasks; however, structured menus which follow a prescribed order are also becoming popular.

The distinction between user-directed and computer-directed tasks is partly one of program branching, but also one of user attitude. Just who or what is directing the course of events may be a matter of opinion. Users may feel that they are in command, or they may get the impression of being passively led down the garden path. Whichever impression dominates will be a function of the user's level of experience and the complexity of the system. Menu systems may increase the complex-

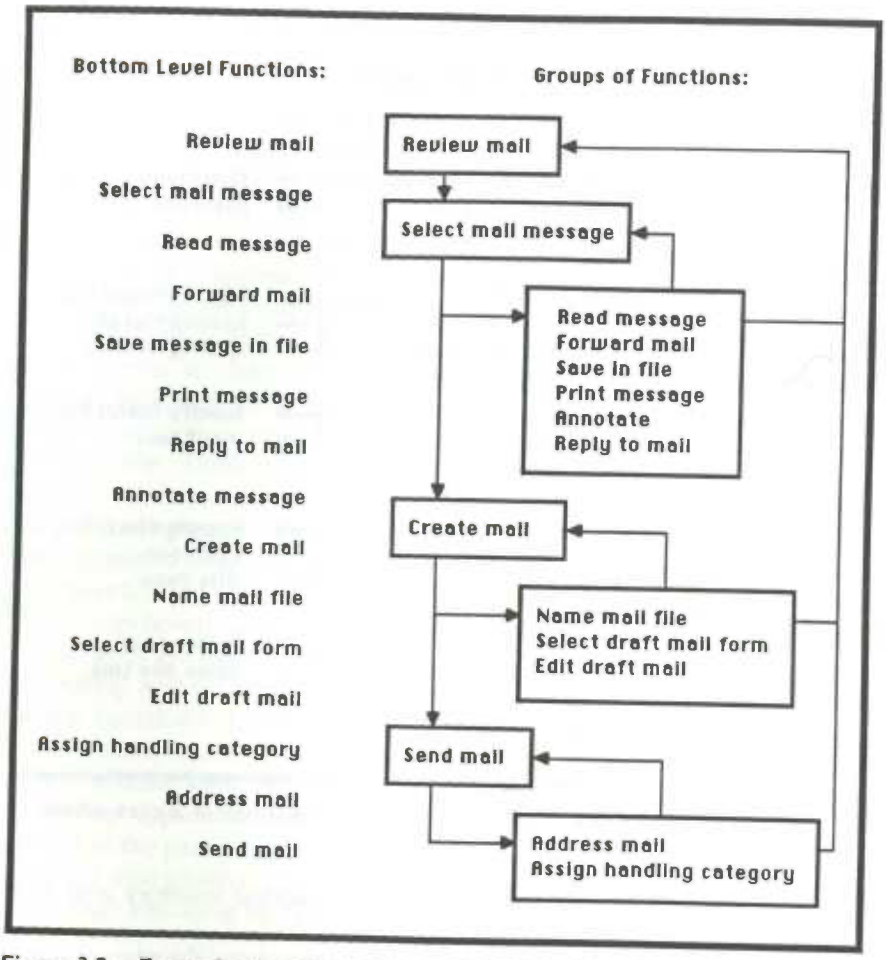

Figure 3.2. Example of a bottom-up analysis of functions in an electronic mail system.

ity of the system by providing breadth of choice at any particular level and by providing successive levels of options. The user's impression may be one of great choice latitude, flexibility, and power. On the other hand, a limited number of menu items may result in a feeling of constraint, inflexibility, and powerlessness. The user's impression, however, is relative and depends on the match between the degree of latitude desired and that provided. It will be argued in the next section that systems may err in providing too little or too much flexibility. A system with scores of superficial options may appear very powerful on the surface but lack true usability. Alternatively, a system that appears

very simple may prove to be efficient. But a system that is too limited may require added work on the part of users.

Another issue related to user- versus computer-directed tasks is that of switching control between the user and the computer. Typically, human/computer interaction is a dialog in which control is passed back and forth between the user and the computer. The user issues a command, the computer executes a procedure and informs the user, who in turn evaluates the result and responds with another command. For users to interact effectively in such an environment, they must know three things: (a) who's turn it is; (b) given that it is the user's turn, what the currently valid set of commands or appropriate input is; and (c) given that it is the computer's turn, what the computer is currently doing or what the result will be. Techniques used in menu selection help to provide some of this information. Menu prompts indicate when the user has control to select options. Menus display the currently available set of options, reducing the probability of selecting an invalid command. Additional feedback is required to inform the user of computer operations and results once a menu selection has been made. To the extent that this information is available and meaningful, the user will be able to operate efficiently. When it is lacking, users will experience confusion and loss of control while performance will be slow and error prone.

3.3 A THEORY OF COGNITIVE CONTROL

The primary function of the human/computer interface is to provide users with an efficient means of controlling a complex process. In the same way that one would exercise control of an automobile, the user controls a computer. The operator of an automobile monitors displays (from instruments on the dashboard to the scene outside the windshield) and manually adjusts controls (from the steering wheel, accelerator, and brake to the light switches, turn signals, and radio). Control of computer processes, of course, differ in a number of ways. Computer processes are generally more autonomous in the sense that they are started and left to run on their own; whereas, automobiles require continuous close control (Wickens, 1984). Control of computer processes is primarily digital rather than analog although increasingly analog input devices are being used. Finally, computer processes vary extensively in their level of complexity (Card, Moran, & Newell, 1983). Control of the process may be as simple as turning on or off a switch or as complex as controlling a simulation of the universe.

Control theory deals with the interrelationships among ongoing processes, informative feedback, and control mechanisms. In general, one

is interested in using control theory to optimize efficiency, productivity, and/or quality. Similarly, in human/computer interaction, the designer is interested in optimizing some aspect of performance. The challenge is considerably greater than optimization of a purely mechanical process since the human is a prime source of control in the process. The user is not only a part of the control loop, but he or she typically maintains a higher strategic level of control. This *metacontrol* may be thought of as control of the control process and may involve such functions as (a) initialization, (b) parameter specification, and (c) programming of the control process itself.

Menu selection is an attractive mode of human/computer interaction in that it explicates interrelationships of the control process in terms of a menu structure. The list of menu items and succession of menu frames help to reveal the underlying states and transitions of a program to the user. Menu selection is a particularly attractive mode of control when it allows users to exercise metacontrol over computer processes. Such systems may allow users to design their own menus, program macro-menu actions via direct manipulation, and organize files into menus using hierarchical file servers. These types of metacontrol will be discussed in a later chapter.

In Chapter 1, a model of human/computer interaction was presented in Figure 1.5. The central part of the figure is the overlap of the circle (cognitive processes) and the square (computer processes). Control involves information output from the computer (top left-going line) and commands from the user (bottom right-going line). A theory of cognitive control, however, must entail more than directions of information flow. In this section the issue of a system function will be addressed. Listed below are the main tenets of a theory of cognitive control dealing with system functionality.

The Apparent System Complexity/Functionality Should Match User/Task Need. To maximize user efficiency, the complexity of the system functionality should match user need. Dehning, Essig, and Maass (1981) assert that for optimal performance and the human/computer interface, the system must provide maximal flexibility while at the same time minimize subjective operating complexity. In addition to the consideration of complexity, the present theory emphasizes a match in functionality between the system and user needs.

To express this more formally, let F_S be the set of active system functions, and let F_U be the set of functions required by the user to perform a set of tasks—T. In general, system functions do not directly perform all the possible tasks. If they did, one would need as many functions as there are tasks. A statistical package, for example, would not have a function to calculate the mean of 2 numbers and another to calculate the mean of 3 numbers, 4 numbers, and so on. Rather, the

package would include another function to let the user set the sample size. At the other extreme, a system may be developed with a minimum number of functions capable of performing all of the desired tasks. A six-function calculator could be used to perform all of the statistical calculations needed. However, to do so would require a lengthy application of functions on the part of the user. Cognitive control is facilitated when the user's conceptualization of functions required to perform a set of tasks matches the set of system functions—$F_S = F_U$.

To the extent that $F_S < F_U$, performance will decrease because the user will have to work around the system's lack of functionality by repeated application of existing functions. To the extent that $F_S > F_U$, performance will also decrease because the user will have to sort through a number of superfluous functions to locate needed functions. With experience, users may learn to overcome these problems. However, the learning time and ongoing cognitive overhead may be too much of a burden to bear.

A second problem in matching functionality has to do with the structure of the menu. The menu which implements system functions may be overly complex or impoverished. Menus are overly complex when scores of infrequently used items are constantly displayed. Well-designed hierarchical menus may direct the user to frequently required items, while at the same time allowing access to highly specialized items. Menus may be impoverished even though they contain all of the necessary items if they allow only a limited set of paths to those items. Impoverished menus require the user to spend an inordinate amount of time traversing the menu to get to the needed items.

Functional complexity and menu complexity may be thought of as two somewhat related dimensions of a system. As shown in Figure 3.3, the ideal system exists when the complexity along both dimensions matches that of user need and proficiency. To the extent that the system is off center, performance will decrease. When menus are inadequate, users must contend with inefficient paths; whereas, when they are overly complex, users must contend with more menu items per frame. Moreover, when functional complexity is inadequate, users must select more functions to accomplish a task; whereas, when it is overly complex, users must search through more functions to find the appropriate one. Proper positioning on these two dimensions is the key to matching software to tasks.

The System State Transition Diagram Should Optimize the User Subtask Transition Matrix. A state transition diagram specifies a set of states (indicated by circles), the possible transitions from one state to another (indicated by arrows), and the conditions under which a transition occurs. In menu selection systems, the allowable transitions are determined by the menu structure and the conditions for transitions are

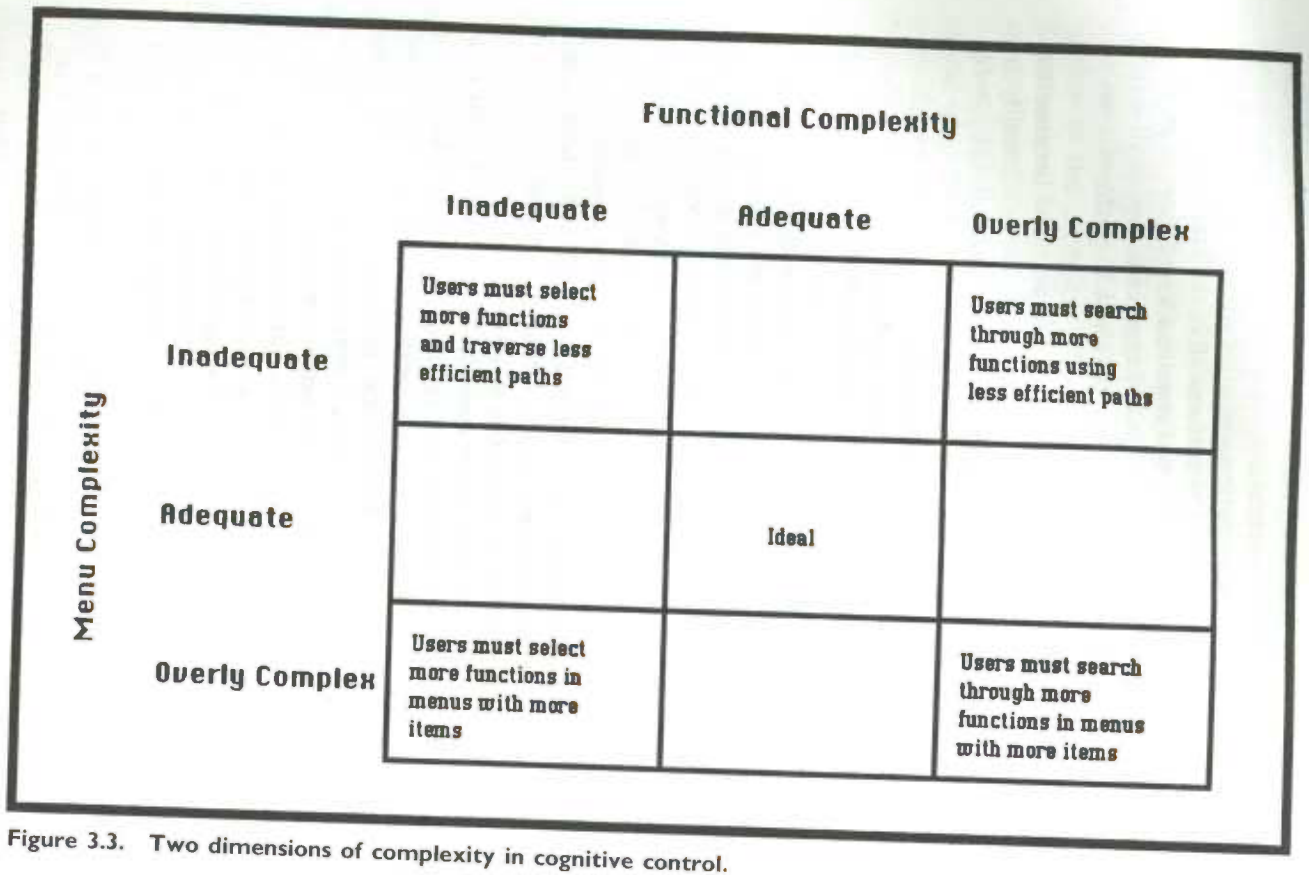

Figure 3.3. Two dimensions of complexity in cognitive control.

determined by user selection. To perform a task, users call upon a number of subtasks. Given the set of subtasks, a transition matrix specifies the probability that one subtask follows another.

The upper panel of Figure 3.4 gives an example of a probability transition matrix for a set of subtasks for playing a computer game. For each subtask implemented at state n (shown in the rows), there is a set of probabilities that the user needs to go to for particular subtasks at state $n+1$ (shown in the columns). For example, if the user is at subtask A, there is a .5 probability of going to subtask B and a .5 probability of

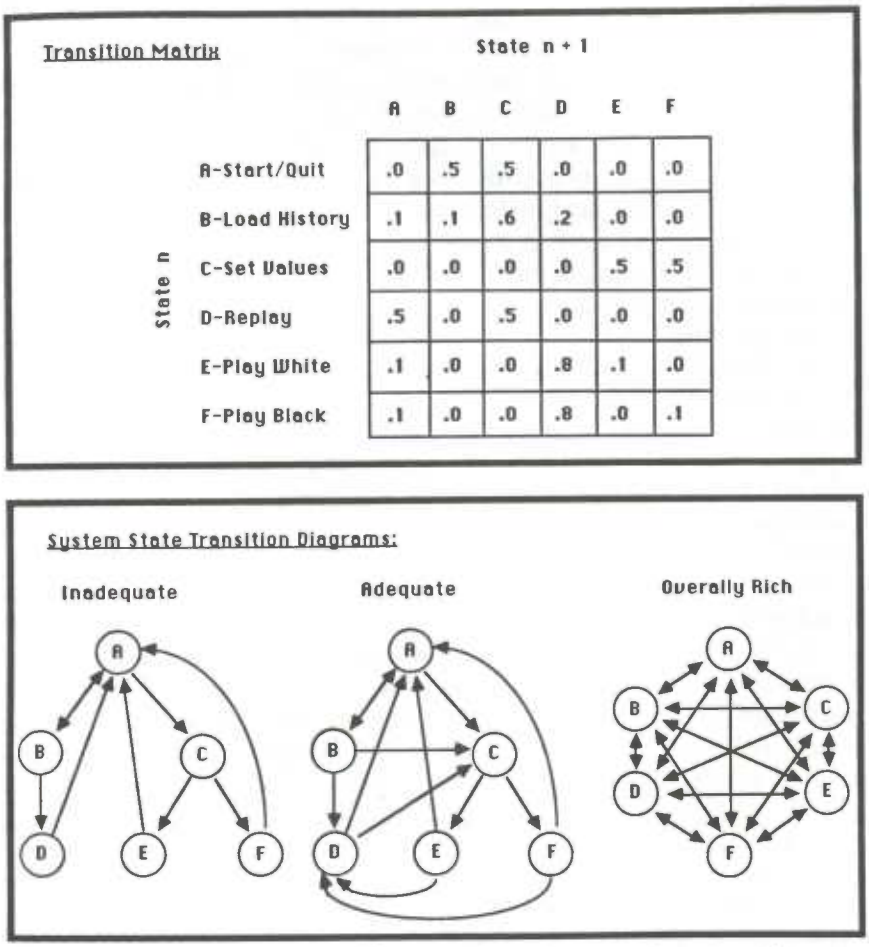

Figure 3.4. User subtask transition matrix and system state transition diagrams representing inadequate, adequate, and overly complex menu structures.

going to subtask C. Given this probability matrix, one may evaluate the graph structure of a menu system which implements these states. A good menu should allow direct paths for high-probability transitions. On the other hand, low-probability transitions need not be direct, but may require successive selections.

Menus that control such state transitions may be either impoverished in the paths that they provide, or they may be too rich by including a number of seldom used paths. Impoverished menus are inefficient because they require the user to make longer traversals of the menu system. In doing so, the user must make more decisions, selections, and planning. The graph in the bottom left of Figure 3.4 illustrates a hierarchical menu from the root node A to terminal nodes D, E, and F with transitions back to the root node. Although it contains 13 transitions, including transitions to previous nodes indicated by double arrows, it does not implement several important high-probability transitions as seen in the matrix: B → C, D → C, E → D, and F → D. Lateral transitions are not implemented; hence, the user must return to the root node and proceed down to the next node in the sequence. The menu graph at the bottom middle of Figure 3.4 provides a better match with the same number of transitions. A number of unnecessary paths up the tree are eliminated and the high-probability lateral transitions are incorporated in the structure.

The degree to which a menu structure is impoverished can be estimated from the probability transition matrix. For a given number of subtasks that have to be performed, one can estimate the percent of the expected number of transitions required by an impoverished menu relative to a menu that implements a path for all non-zero transitions. For example, the percent of extra traversals for the "inadequate" menu in Figure 3.4 is 60% over that of the "adequate" menu.

On the other hand, menus may be overly rich when they include a number of superfluous or underutilized paths in the structure. An overly rich menu is shown at the bottom right of Figure 3.4 which allows a transition from any node to any other node. Although a rich menu minimizes the number of frames traversed, the increased number of alternatives may retard choice and response times of the user. Impoverished menus will evidence faster response times per frame, but the overall time to perform a task may be longer since more responses are required. Designers must tradeoff the costs between rich and lean menus to optimize performance.

The efficiency of menus can be compared by estimating expected user response time across the menu. Response times are a function of the number of alternatives in each frame. For the present, assume that the response time is given by a power law—$R = n^p + 1$—where n is the number of items per frame and let $p = .7$. Table 3.1 gives the response

times at each node of the three menu graphs shown in Figure 3.4. Using the transition probabilities also in Figure 3.4 and the graphs, the expected response times were calculated and are shown at the bottom of Table 3.1. The expected response time for the "inadequate" menu is much longer, due to the fact that 60% more responses are required for the desired state transition. The "overly rich" menu also shows a higher response time due to slower responses at each frame. It can be seen that changes in the menu structure can result in rather large changes in performance. Similarly, designers should compare alternative implementations of menus or iteratively vary menu structures to find one that optimizes performance.

3.4 FUNCTIONS OF MENU SELECTION

In this section the specific functions of a menu selection will be discussed. Each selection made by the user in a menu performs one or more functions. The function may be merely to branch to another menu or it may be to do something else. This section will define the four functions of (a) pointing, (b) command control, (c) output, and (d) input. Other taxonomies of menu function could be specified based on program operation; however, the present one will prove useful from the perspective of understanding cognitive control by the user.

3.4.1 Pointing: Moving to a New Node

Each selection by the user branches to a successive node in the menu tree. Although this is an inherent function of all menus, the importance of this function varies among systems. In some cases, it is the sole purpose of the selection to traverse the menu tree. Selections serve in a stepwise specification of an object or command. The overriding purpose

Table 3.1. Expected User Response Times for Three Menu Types.

	Inadequate	Adequate	Overly Rich
A	2.62	2.62	4.09
B	2.62	3.16	4.09
C	3.16	2.62	4.09
D	2.62	2.62	4.09
E	2.62	2.62	4.09
F	2.62	2.62	4.09
Expected Response Time	4.24	2.64	4.09

is one of pointing to something. From the perspective of the user, menu selection is like navigation. The sole purpose of each selection is to steer the system toward a destination. In this way the process is goal-oriented. Although selection errors take the user off track and lead to a loss of time, they are not destructive and are in general undoable.

The pointing function is essential in hierarchical menu systems. An example would be a menu system for a timesharing utility. A number of services are available to the user; but in order to access one, the user must specify its name. Rather than having to type in the name of the service, the user is given a series of choices. The services are typically clustered by type into a hierarchy. Each selection by the user does nothing except move to the next level of specification. Once the user comes to a particular service on the system, the function of the menu changes from traversal to implementation.

3.4.2 Command Control: Executing a Procedure

The second function of selection is to direct the computer to execute a procedure or to implement some action. In this case, the choice evokes a procedure that performs some function. It may open or close a file, write or read to a file, transform data, and so forth. As noted, this may only occur after the user has traversed the menu tree and is at some terminal node. On the other hand, every selection made by users may initiate a program procedure as they traverse a menu network. In this case, successive selections evoke a sequence of procedures. The users determine the selection and order of procedures. From the user's perspective, selections are "go-ahead" commands for the system to execute a function. It should be clear to users that selections are not merely pointing to items, but that they are activating changes. Errors in command control may not be so inconsequential as in pointing to the wrong node. Consequently, the user needs to be informed if selections are undoable before they are selected.

An example of command control would be a telecommunication program with menu selection. If the user selects the option PHONE, the system executes a procedure to dial and connect to a remote system. The user may then select RECORD to open a file and save the text of the interaction. Later the user may select STOP RECORDING to close the file and HANG UP to disconnect from the system.

3.4.3 Output: Displaying Information

The main function of menu selection may be to display information. The user may be searching for a specific piece of information in a database, or merely browsing through information contained in a menu hierarchy.

The output function is very similar to the pointing function of menu selection except that from the perspective of the user, attention is focused on the content of the output information rather than on the pointer to it. Like the pointing function, the output function is nondestructive. If the user gets off track, he or she may loose time but should be able to recover from selection errors. Information search may be either goal-directed or casual browsing, but the path taken will be determined by the content material and the user's interests. Consequently, users will spend considerable effort processing output information in order to evaluate subsequent menu items.

Online technical manuals provide a good example of the output function. A particularly good example is the interactive encyclopedia system (HyperTies) developed by Shneiderman and his associates (Shneiderman & Kearsley, 1989) which displays articles containing highlighted words or phrases. If the user selects an item, the system displays a definition and/or a related article. The user may then browse through a set of interrelated articles.

3.4.4 Input: Data or Parameter Specification

The last function is one of input. Each selection by the user specifies a piece of data to be input into the system or the desired value of some parameter. Input values may be indicated by single menu selections or by traversing a menu hierarchy. In the latter case, traversing the menu specifies an input value incrementally. Although this type of input function is again similar to the pointing function, the emphasis from the perspective of the user is on input of information to the system. Menu selection adds structure to data input by limiting input to allowable values and reduces errors due to incorrect syntax and typing. In general, selection errors can be easily corrected if detected. Nevertheless, undetected errors remain a problem, as usual, in data entry.

3.4.5 Four Menu Functions in Perspective

It should be emphasized that this fourfold taxonomy of menu selection should be interpreted from the perspective of the user. In terms of the code driving the system, it is no doubt meaningless. In actuality the code may be written so that for every selection (a) a new frame is pointed to, (b) a procedure is evoked, (c) information is displayed, and (d) data is input. The main distinction among these functions is not so much in what the software does but in what the user thinks it is doing.

Furthermore, these functions apply most directly to individual menus or sequences of menus in a system; whereas, most menu systems are

mixtures of these functions. Parts of the menu tree may exist merely to branch to some procedure; other parts of the system may exist for the purpose of displaying information; and still other parts may provide menus of executable procedures.

Having said this, it is also true that systems may emphasize only one function. Menus for timesharing systems, file management, and document retrieval systems lean more toward traversal of a database via the pointing function. Menus for operating systems and application programs, such as word processors and spread sheets, lean more toward executing procedures via the command control function. Menus for help systems, tutorials, and information retrieval systems lean more toward displaying information via the display function. Finally, menus for online questionnaires and data entry systems lean more toward data and parameter specification via the input function.

The way in which a menu is designed should be commensurated with the perceived task and the way in which the system is used to accomplish the task. MacGregor, Lee, and Lam (1986) distinguished between "command" menus and "videotext" menus, and noted that command menus are designed to select a relatively limited set of items (30 to 100). Experienced users have learned the positions of menu items. On the other hand, videotext menus are likely to access relatively large databases with thousands of documents requiring a large number of menu frames. In command menus the user generally does not read all of the items but merely scans to the desired one and selects it. In videotext menus the user may need to read each item carefully in order to determine the next selection. While command menus may be designed with a relatively large number of options, videotext menus, according to Paap and Roske-Hofstrand (1986), should focus the user down a narrower path by presenting a smaller number of items distributed across more menu frames.

Menus that merely serve to point successively to other frames should be designed to facilitate traversal, possibly by jump-ahead commands, and they should provide the user with a sense of position, possibly by displaying a path or a global map of the menu tree.

3.5 OPERATION BY MENU SELECTION: COMMAND MENUS

Many applications use menu selection as a replacement for command languages. Rather than specifying a command by entering it via the keyboard, the command is selected along with its operand via menu selection. Command menus have the advantage of overcoming two inherent problems of command languages. First, menus help to disambiguate terms that can have multiple meanings. A term such as *list* may

be a command to display a list of items, or it may be the name of an object. The particular meaning of a term can be determined by the context. The second problem is that a function could have a number of equally likely names. Menus help to resolve which of a number of synonyms is recognized by the system as a command. For example, the function of terminating a process may be labeled: stop, halt, quit, end, fin, bye, kill, term, and so on. With a command language, one must remember which term is appropriate. Natural language systems may recognize a number of synonyms; however, users must still go through the cognitive process of generating a term and wondering if it was correctly recognized. Further difficulty arises if two or more synonyms result in different actions by the system. For example, "quit" may exit a program and return to the operating system, "halt" may stop the processor until a continue command is used, "kill" may terminate a process without storing the results, and "end" may gracefully terminate a process and store the results. By listing all four commands as menu items the user is alerted to the fact that each has a different effect.

Menus for control of operations usually involve a sequence of selections from (a) a set of commands, (b) a set of operands, and (c) a set of options or parameter values. Commands are essentially synonymous with verbs, operands with names of objects, and options with qualifiers, such as adjectives and adverbs. The sequence of selections to perform a task constitutes a sentence which must conform to the grammatical order required by the system. For example, the command to open a specific file could be implemented by first selecting the option "Open" from a list of operations and then selecting a file from a list of names of files. Depending on the application, a different order of selections for the command, the operands, and the options may be required. A document (operand) may be selected first, then the command (print), and finally the print options (e.g., draft, letter quality). On the other hand, the application may require that the print options and the document be specified prior to the print command. For command languages, research suggests that the command precede the operand (Barnard, Hammond, MacLean, & Morton, 1982; Barnard, Hammond, Morton, Long, & Clark, 1981).

On the other hand, with menu selection and particularly with direct manipulation, the preferred order may offer operand first. In this order, a file would first be selected from a list, then a command (copy, delete, duplicate, etc.) would be selected. The order in which selections are made may have a number of implications. When commands, operands, and options are considered, there are six possible orders to consider. Table 3.2 lists these orders and some considerations for each.

All possible orders can be implemented with menu selection. The system may impose a fixed order or allow the user to select commands,

Table 3.2. Possible Orders of Selection of Commands, Operands, and Options and Their Implications.

Order	Implications
Commands-Operands-Options	Actions are planned first. Operands are thought of as objects of commands. Options refine commands as applied to selected operands.
Commands-Options-Operands	Actions are planned first. Options refine commands irrespective of operands or preconditionalize operands. Operands are thought of as objects of commands.
Operands-Commands-Options	Targets are planned first and are thought of as the subjects of commands. Options refine commands as applied to selected operands.
Operands-Options-Commands	Targets are planned first. Options refine operands or preconditionalize commands. Commands are thought of as action verbs applied to the targets.
Options-Commands-Operands	Options preconditionalize commands and operands before they are selected. Operands are thought of as objects of selected actions.
Options-Operands-Commands	Options preconditionalize commands and operands before they are selected. Operands are thought of as subjects of commands.

operands, and options in any order. The particular order used has implications about the way in which the user is thinking about the task and forming a plan to be executed. In many cases the selection of commands, operands, and options may be left at a default or preset value. The prior specification of options requires that the user plan ahead, but has the advantage in that he does not have to process the option menu each time a command is made. However, if it is likely that the values change or that incorrect options could lead to serious problems, it is advisable that the option menu be presented following command selection. Prior specification of operands allows the user to apply a series of operations to the same set without repeatedly specifying its members. A user may select several files (operands) and the command to print and then the command to delete. Both operations apply to the same set of files.

When the number of commands is large, they are often presented in a hierarchical structure. Commands may be grouped by type. For example, commands may be grouped that have to do with file operations, resource allocation, account management, and so forth. When commands must be selected by a series of menu choices, two problems arise. First, the top menu does not list commands, but only labels standing for the sets of commands. Sometimes these labels are not very informative about the commands they represent. For example, the label "Goodies," which has been used in some packages to represent a mixed set of

commands that do interesting things, is provocative but less than informative. The second problem is that it takes more than one selection to locate and execute a command. Experienced users may be able to type the command name faster than making a series of choices. However, if the menu hierarchy is broad and system response time is fast, menu selection has the potential of resulting in faster execution of commands rather than typing command names on the keyboard.

System operations are generally much more complicated than just making selections. One command may change the state of the system such that subsequent commands have different effects or such that the system enters a different mode that implements different commands. For example, users face one set of commands when they are working with the operating system and a different set when running a particular application program. The availability and effects of commands change drastically depending on the environment. One of the major advantages of menus over command language is that the menus serve to provide information about the current program state. Since users frequently have to move from one mode to another, explicit lists of commands in the form of menus have the additional advantage in such situations by restricting the allowable set of commands to those that are currently active. Menus may display only the allowable commands, or they may display the full set with inactive items grayed out as shown in Figure 3.5. The advantage of retaining the grayed-out items in the list is that the

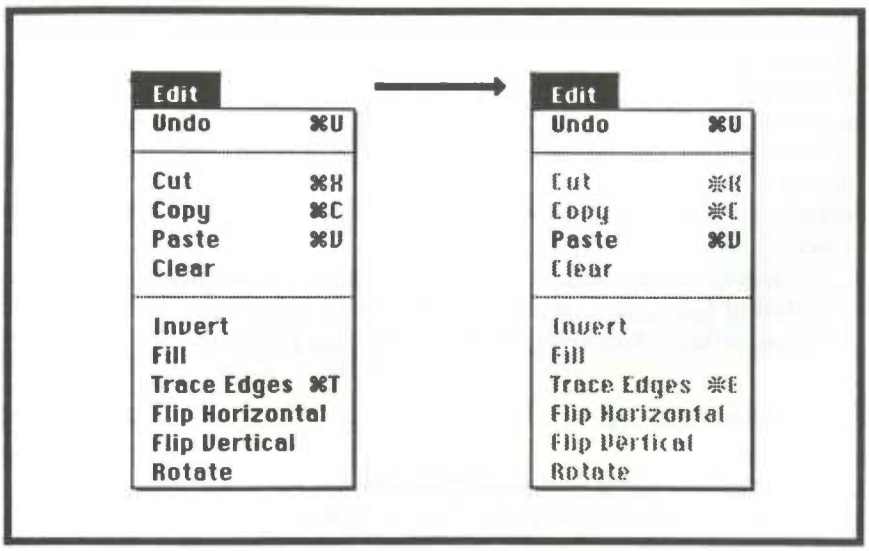

Figure 3.5. An example of graying-out menu options when they become inactive (MacPaint®).

presence and location of items are fixed. Users are exposed to all the items and are informed as to which are current. What they may not know, however, is how to switch modes so as to activate desired items.

Items may also be switched or rephrased depending on the current state. In one state the menu may display the item "Open File." In another state, it may change the item to "Close File." Or it may display both "Open File" and "Close File" with one or the other grayed out. The first form allows for shorter lists, but may confuse even the experienced user who must carefully check the current state of the menu before selecting items. When menus switch options depending on context, it can be frustrating to the user who is perplexed about the cause-effect relationships between program state and menu lists. An interesting example is provided by MicroSoft-File™ in which the pull-down file menu has three states, depending on whether the user has opened a file, a report, or a form (see Figure 3.6). The spreadsheet program Excel™ handles this problem in a different way by providing a generic open command with a subsequent menu for specifying a spreadsheet, chart, or form (see Figure 3.7).

3.6 DECISION BY MENU SELECTION: DECISION MENUS

Menu systems can effectively be used to assist users in making decisions. Decision making is a complex task when the number of competing alternatives is fairly large and/or when the alternatives are composed of many positive and negative consequences. When the decision can be structured as a series of sequential choices, menu selection provides an appealing method of simplifying the task. Such systems present a series of choices that help formulate the decision. The decision may be as simple as which help file to access, or as complex and critical as deciding among alternative courses of action to rescue a stranded vehicle in outer space.

Decision systems may be implemented using menu selection in several different ways, depending on the formal properties of the decision and the method of handling decision trees and matrices of alternatives.

3.6.1 Decision Trees

One way of handling complex decisions is to use decision analysis to decompose a complex decision into a hierarchical decision tree. The decision tree represents each sequential choice as a node in a graph. Alternative actions on the part of the human are represented as arcs pointing to other choice nodes or to states of the world. Alternative

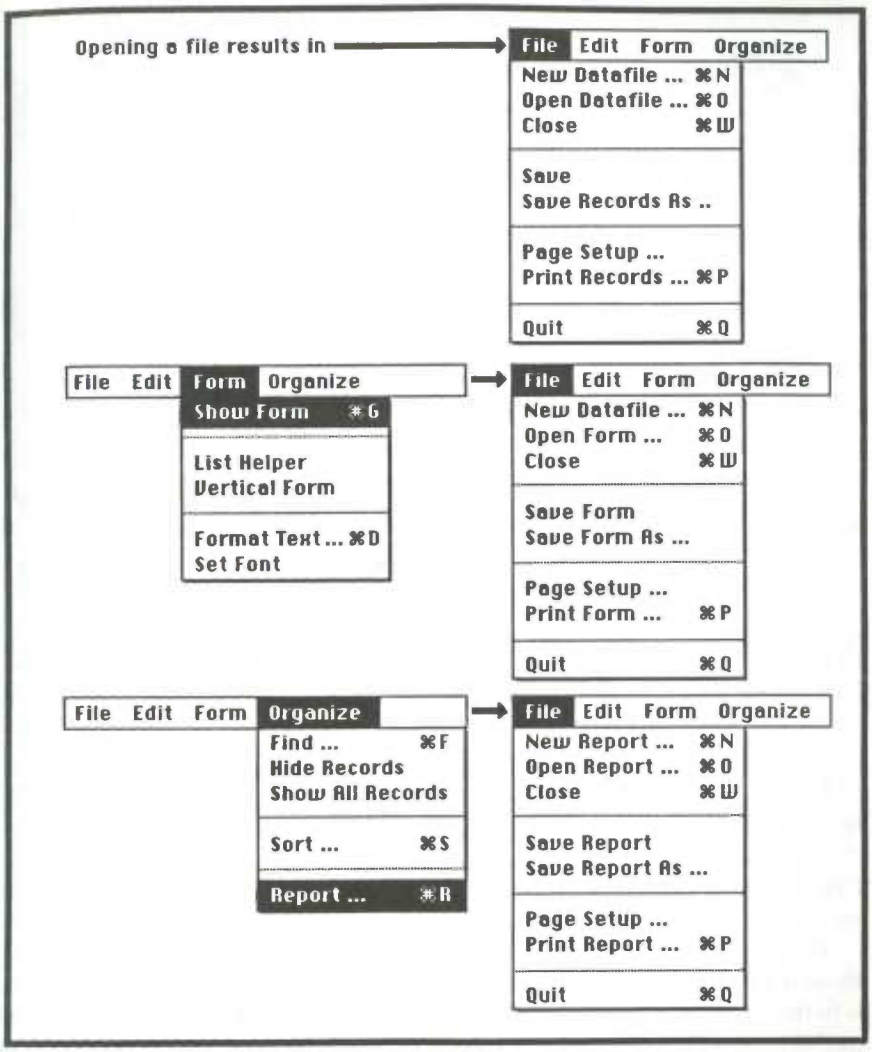

Figure 3.6. Changes in pull-down menus caused by other menu selections in MicroSoft™ File. The File menu has three states depending on whether one has opened a file, a form, or a report.

consequences in the environment are also represented by arcs leading to subsequent choice nodes or states of the world. In a sense, decision making then entails traversing the decision tree by making a selection at each choice point. The advantage of implementing the decision on a menu system is that choices are laid out at each level and the user can

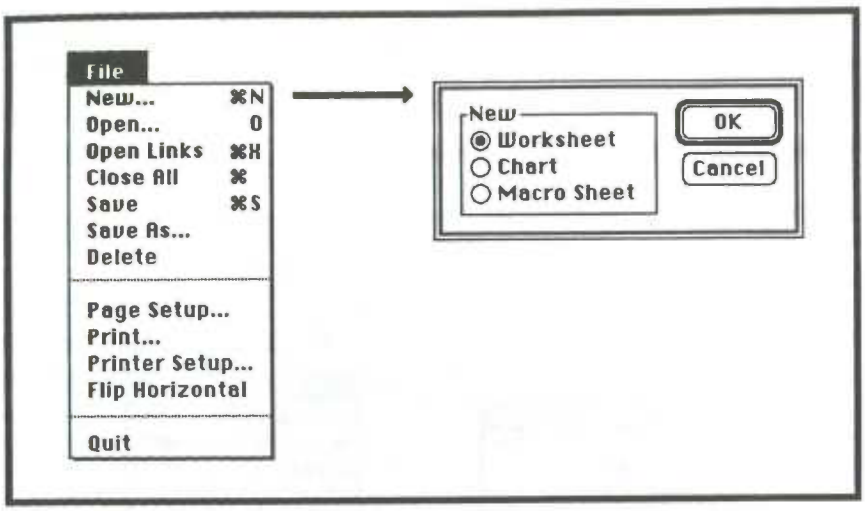

Figure 3.7. An example of adding an additional level of menu selections in MicroSoft™ Excel to specify the type of file.

easily traverse the decision tree by making selections. However, the shortcoming is that the decision maker may focus on only one choice node at a time and ignore more remote consequences. In general, the goal of decision analysis is to provide a more comprehensive analysis in order to locate an optimal terminal node. To some extent menu systems may avoid this problem by displaying one or several plies down the decision tree. However, when extensive search through the decision tree is required, other approaches are called for, such as Bayesian analysis or tree-searching techniques.

When menu selection attempts to implement a decision tree, it should be evaluated on how well it captures the decision process and whether it has hidden consequences. It is easy to misrepresent the decision space in the menu tree and lead the user down a "garden path." Figure 3.8 gives an example that is not atypical of many such applications. The extremely complicated decision of career choice is laid out in terms of interests and abilities. Such systems may be provocative and even instructive in the way they elucidate a number of decision factors. But they are often misleading in that early choices have unanticipated consequences on subsequent options. Although users may be ambivalent about early choices, they are nevertheless forced to commit themselves to one path, thereby eliminating other paths that would have led to more desirable options. Such systems have the potential of actually degrading decision-making ability. However, properly used,

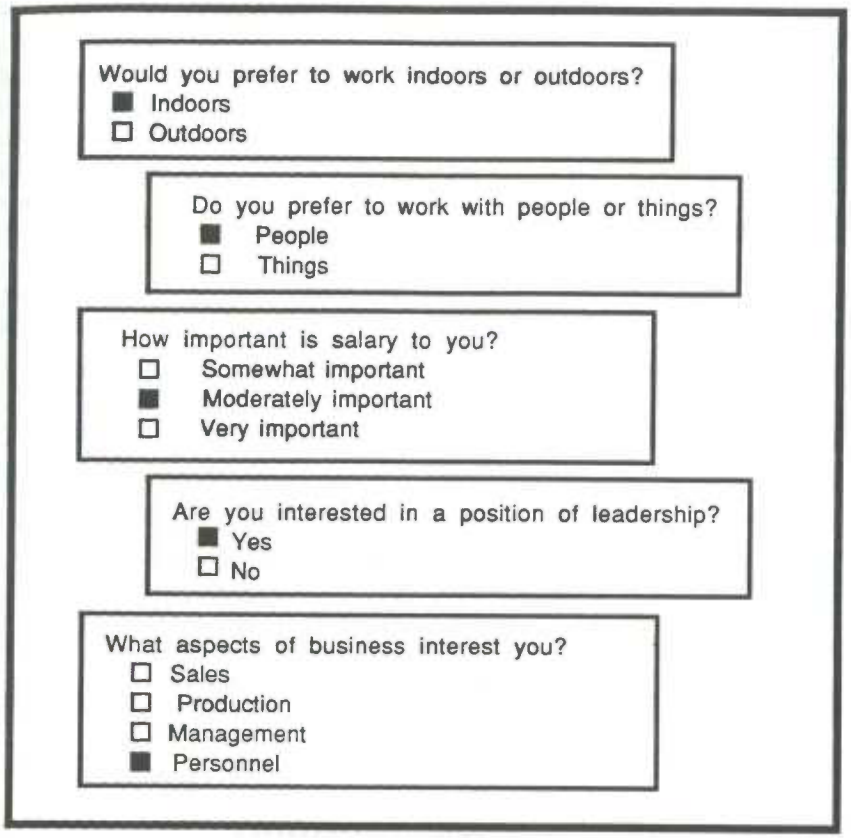

Figure 3.8. An example of using menu selection in the decision-making task of career choice. (Note that the order of selections, choice structure, and completeness of the tree can lead to substantial differences in choice.)

menu selection may capture the decision sequence in a way that explicates a complex decision process in a straightforward and complete manner in order to facilitate the decision process. In general, one would expect menu selection to be most applicable when the structure of the decision tree is well-known or clearly determined by a prescribed rule.

3.6.2 Decision Matrices

Many decisions involve a selection among a complex set of alternatives. Decision theory provides a formal method of representing alternatives as sets of attributes. Alternatives can be thought of as multidimensional vectors of attributes. Consider running shoes, for example. They may be

described by the attributes of price, weight, midsole construction, outsole construction, type of last, and heel lift. In turn, physical attributes result in subjective preferences along the attributes, such as overall comfort, shock absorption, flexibility, toe room, overall stability, heel counter, sweat loss, and so on. Table 3.3, for example, lists 15 running shoes with their price, and judge's ratings on seven attributes. The decision task is difficult even in this simple case of deciding on running shoes. Decision theory in conjunction with menu selection, however, can simplify the task considerably.

Decision theory operates on the assumption that a quantifiable utility measure exists for each attribute. Each alternative i can then be described by an array of numbers:

$$A_i = \{u_{i1}, u_{i2}, u_{i3}, \ldots u_{ik}, \ldots u_{iK}\}.$$

Decision making then involves a process by which attribute utilities are compared and combined in order to determine the preferred alternative. Decision theory has investigated a number of possible methods for doing this (Montgomery & Swenson, 1976). A few of these are outlined below. Each requires a different level of assessment of preference and makes different assumptions concerning how to integrate the relevant attributes.

Ordinal Dominance Rule. An alternative is selected if for every relevant attribute the decision maker prefers that alternative, or at least does not prefer another alternative over that one. To implement the prefer-

Table 3.3. Object-Attribute-Utility Matrix for Running Shoes

Brand and Model	Price	A	B	C	D	E	F	G
New Balance 470	$53	5	4	5	5	3	3	2
Asics Tiger Epirus	$80	5	5	5	4	4	4	5
Turntec Quantum Plus	$60	4	4	5	5	4	4	4
Saucony Shadow	$59	5	5	5	4	2	4	5
Asics Tiger Ultra 1000	$57	5	4	5	4	3	4	5
Saucony America	$48	4	4	4	4	3	4	5
Brooks Trilogy	$70	5	4	4	4	3	3	3
New Balance 1300	$130	4	5	4	4	4	4	3
Adidas ZX500	$75	4	4	3	3	5	4	2
Brooks Chariot	$62	3	3	3	4	5	5	3
Reebok DL5600	$56	4	4	5	4	3	3	4
Nike Venue	$60	3	3	3	4	4	4	4
Nike Vortex	$60	3	3	4	3	3	4	3
New Balance 575	$66	3	4	3	4	4	4	4
Etonic Mirage	$45	3	3	4	4	2	2	4

Note: 5 = better, 1 = worse, A = overall comfort, B = shock absorption, C = flexibility, D = toe room, E = overall stability, F = heel counter, G = sweat loss.
From "Running Shoes," *Consumer Reports*, October 1986, pp. 654–655. Reprinted by permission.

ence rule in menu selection, the system would present a menu for each attribute listing the available levels. Decision makers would select the preferred level for each and the system would identify the alternative(s) possessing those attributes. Unfortunately, it is usually the case that such ideal combinations do not exist and no alternative completely dominates the rest. In Table 3.3 for example, there are no running shoes that are the least expensive and have the highest ratings on all the attributes. Consequently, one must in some way relax the requirements.

Minimum Criteria Rule. An alternative is selected if it meets or surpasses a minimum criteria on each attribute set by the decision maker. This rule would again be implemented by presenting menus to the decision maker for each attribute listing the available levels. But instead of selecting only the preferred level, the decision maker selects all acceptable levels (e.g., price less than \$60, overall comfort > 3, etc.). If no alternatives are located, the decision maker may iteratively relax the criteria until an alternative is selected. If a number of alternatives meet the criteria, the decision maker may raise the criteria.

The minimum criteria rule is similar to the "satisficing" rule discussed by Simon (1976) in which the decision maker selects the first alternative that meets a set of minimum criteria. The difference is that the satisficing rule is self-terminating; whereas, the minimum criteria rule requires that the system perform an exhaustive search through the entire set of alternatives.

It is usually the case that some attributes are more important than others. Neither of these two rules takes this into consideration. An alternative may be dropped because it does not meet a criterion on a relatively unimportant attribute. The remainder of the rules below take into consideration attribute importance.

Lexicographic Rule. An alternative is selected if it tends to be preferred over other alternatives on attributes judged to be of greater importance than others. To implement this rule the system would first present a menu of all of the attributes and the decision maker would select the most important. Then the levels on that attribute would be presented and the decision maker would select the preferred level. The system would then list the remaining attributes and the decision maker would select the next most important attribute. The levels on that attribute would be listed and the decision maker would select the preferred level, and so on. For example in Table 3.3, overall comfort might be selected first, and a rating of 5 preferred. Price might be next in importance, and among those having a rating of 5 on overall comfort, the price of \$57 is preferred. The decision process would stop there since there is only one pair at that price.

Elimination by Aspects. An alternative is selected if it remains after the decision maker has sequentially eliminated alternatives not possess-

ing desired attributes. A series of studies by Tverski (1972) suggested that in many situations people make decisions through a process of elimination. The decision maker considers one attribute at a time. Alternatives that do not possess a desired aspect are dropped from consideration. Those that possess the aspect are retained. The process continues until all but one alternative remains. The elimination by aspects rule corresponds closely to the process of menu selection. The rule may be implemented in a way similar to the lexicographic rule. The difference is that the rule is to eliminate alternatives not possessing an aspect as opposed to retaining alternatives having preferred levels of attributes.

There are several disadvantages with the elimination-by-aspects rule. Strict adherence to cut-offs can lead to less than desirable outcomes. For example, the first attribute may be price and the decision maker selects the aspect "less than $60." In this case Turntec Quantum Plus is eliminated although it is only one dollar more than Saucony Shadow. A second problem occurs with the desirability of one attribute that is contingent on the level of another. Only through backtracking and extensive exploration of the menu can the user consider tradeoffs between attributes.

Menu selection can be a very effective method of decision making when the order of the aspects reflects their importance and there are no important interactions between attributes (the presence or absence of one attribute negates the value of another). Unfortunately, menu-aided decision making can be misleading (a) if the order of aspects is ill-chosen, (b) if a weak preference eliminates alternatives that have desirable aspects on other attributes, or (c) if strong interactions exist between the values of attributes. When sales and promotion techniques are embedded in menus (e.g., menu selection for online catalog shopping services), the order of aspects may not be in the decision maker's best interest. For example, the shopper may wish to spend no more than $60 on running shoes. However, the menu may commit the decision maker to certain brand names or features first, and present price only near the end of the choice process. At that point, the decision maker may be lured to spend much more than $60.

Several algebraic rules have been proposed to optimize the outcome of the decision process. These rules make full use of the utility values and are used in computer-aided decision systems.

Weighted Utility Rule. Select the alternative that has the maximum sum of weighted utilities. Multiattribute utility theory (Keeny & Raifa, 1976) makes full use of the utilities of the attributes. For each alternative one may calculate an overall utility:

$$U_i = w_1u_{i1} + w_2u_{i2} + w_3u_{i3} \ldots + w_ku_{ik} \ldots + w_Ku_{iK},$$

where U_i is the overall utility for alternative i; u_{ik} is the subjective utility of attribute k on alternative i; and w_k is the weight of attribute k. For example, assume that the weights are 15, 30, 20, 10, 10, 5, 5, and 5 for price and the subjective attributes A through G. Furthermore, let the ratings in Table 3.3 represent the subjective utilities for the attributes, and let the utility for price equal 10/(log price). Then the running shoe with the highest evaluation is the Asics Tiger Epirus.

Cognitive Algebra Rules. Select the alternative that has the maximum value resulting from an algebraic rule that simulates the decision maker's judgment rule. Information integration theory has demonstrated that for different types of judgments people apply different cognitive rules in order to combine attribute values to make an overall assessment (Anderson, 1980; Norman, 1981).

$$U_i = f(u_{i1}, u_{i2}, u_{i3}, \ldots u_{ik}, \ldots u_{iK}).$$

It could be that consumers average the utilities of the attributes A through G and multiply by the subjective utility of price. If this is the case, then the running shoe with the highest evaluation would be the Asics Tiger Ultra 1000.

The weighted utility and cognitive algebra rules require considerable quantitative information from the user in order to make decisions. Functionally, they are much more complex than the other rules and as such are not particularly condusive to menu selection as a mode of interaction. What is important in decision making by menu selection is that the level of menu interaction be appropriate for the level of decision rule. Menu selection is beginning to be used in conjunction with decision support systems, computer-aided decision systems, and expert knowledge systems. In each case, menu selection may used to input information, to set criteria, and to query the system. In some cases, menu selection replaces command language interaction, but often it is used to structure the decision task according to a prescriptive model of decision making. Decision by menu selection represents one of the most challenging applications of menu selection.

3.7 INFORMATION RETRIEVAL BY MENU SELECTION: INFORMATION MENUS

Large databases may be (a) organized as hierarchically nested records of information, (b) indexed by a number of attributes called facets, or (c) linked as a relational database. For example, retail goods in a department store may be organized in a hierarchy from department (e.g., men's clothing, electronics, etc.) to category (e.g., men's suits, televi-

sions, etc.) to subdivisions (e.g., size 38, 21 inch color) down to particular brands and models. An example of a faced database would be a file of photographs indexed by date, photographer, location, and subject. Some attributes might be collections of other attributes. If the subject is *people*, then they might also be indexed by number, group composition, activity, mood, and so forth. Finally, an example of a relational database would be hypertext documents. The original concept of hypertext was to link all written text together in a network such that a reader could travel from one article to another at will. In actual practice, the reader might start with a general introductory article on some subject that mentions more detailed ideas. The reader would then select one of these ideas and access other articles relating to it.

The structure of the database generally dictates the way in which users may search for information. Menu selection provides a particularly good interface for information retrieval since the user is often unfamiliar with the specific organization and content of the database. The user needs extensive prompting as to existing categories, available attributes, and permissible keywords. Because of the size and complexity of typical databases, even experienced users cannot be expected to be sufficiently knowledgeable about their structure. For that matter, one major reason for searching a database is to find information that was previously unknown to the user. Actually there are two different cases. Users may be searching for an unknown goal object that satisfies some preset conditions (e.g., a book on whale song). On the other hand, they may be locating a known goal object in order to access information about it (e.g., the call number of *Whale Song* by Tony Johnston). As users become more familiar with the system, they learn top-level menus and acquire knowledge about the inherent structure of the data base. But unlike command menus, users will rarely, if ever, become familiar with the extensive lower-level menu information except in well-searched local areas.

In general, menu familiarity is a function of the frequency of access and the user's prior knowledge about objects in the database. In turn, frequency of access is a function of user experience and level of menu. Figure 3.9 shows this relationship in terms of familiarity profiles for a selected set of inexperienced and experienced users and for known and unknown target objects as a function of menu information level. These profiles help to predict the amount of time, difficulty, and effort required on the part of users for accessing information. Designers should realize that the greatest difficulty will be at the intermediate levels of menu search for all but highly experienced users of relatively small databases. Consequently, great care needs to be taken at this point in providing help to users.

In many cases users may have less than a goal-directed approach to database retrieval. Indeed, they may be browsing the database to gain

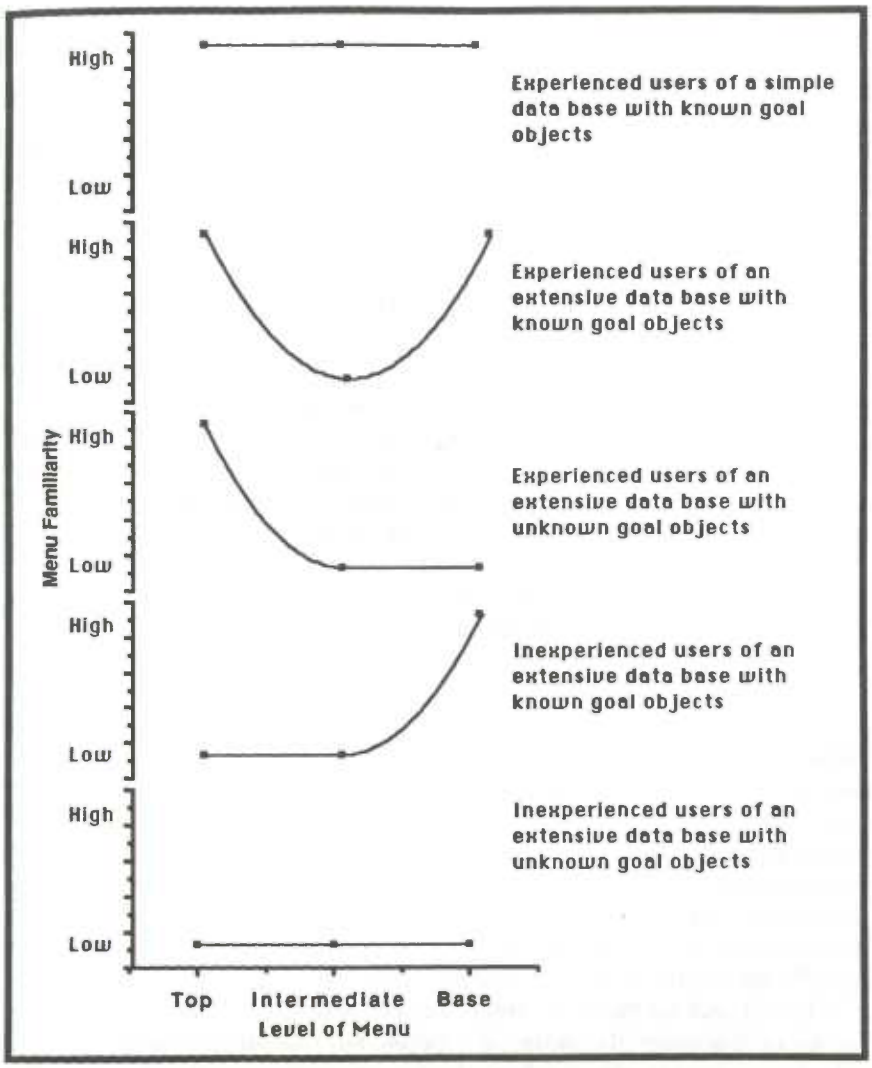

Figure 3.9. Predicted profiles of familiarity with menu information as a function of level in the search for experienced and inexperienced users with known and unknown goals.

an idea of what's there. Menus which list the sets of categories and/or facets quickly reveal the scope of the database. On the other hand, users may be pursuing a serendipitous walk through the pathways of a database. Menus allows for a relatively effortless selection of paths through the system.

Unlike command menus or decision menus, users of information

menus are driven by the information displayed rather than commands required or alternatives desired. This is to say that users are concentrating on the output function of menu selection rather than on the pointing, execution, or input functions. To be sure many information retrieval systems use command menus and decision menus; however, for the actual process of the direct user search of a database, the output function is the most important.

3.8 CLASSIFICATION BY MENU SELECTION: CATEGORY MENUS

Classification is in a sense the inverse of information retrieval. The user has a known object and seeks to place it in a structure or the user specifies the parameters of the object by selecting settings. Menu selection is a natural technique for implementing classification. In particular, classification makes extensive use of the output function of menu selection. At a simple level, a menu may be used to classify cases by gender, income, age, and so on. At a more complex level instances may be classified by the presence or absence of a series of attributes or classified in a hierarchical structure of categories.

The problem of categorization has been the subject of much research in cognitive psychology (e.g., Rosch & Lloyd, 1978; Smith, Shoben, & Rips, 1974). Two different approaches can be taken in classifying instances. One method is to assign instances to categories on the basis of their similarity to a prototype. The prototype is the most typical example of the category. For example a robin is a good prototype of the category *bird*; whereas, an ostrich is not. For a large number of categories, prototypes may be hierarchically arranged in a menu system. The selection of prototypes (labels) and the organization of categories in the database can greatly affect the resulting classification.

The second method of categorization is to assign instances to categories on the basis of defining attributes. For example, an animal is classified as a bird if it has wings, lays eggs, and is warm blooded. The particular attributes used to define categories have a great effect on the resulting classification. Designers may inadvertently bias the user by the types of dimensions used or by the order of attributes in the menu. Early attributes have a tendency to be more salient to the user, setting the context of subsequent choices, and being ascribed as of higher importance than other attributes. The classification scheme used to generate the hierarchical structure sets an agenda for the user. This agenda may not be in the best interest of the user. Instead it may be designed to lead the user to commit to choices that he or she would not have selected under other situations.

The ordering of prototype categories or defining attributes may be set according to a number of concepts.

Scope. Higher levels in the hierarchy may pertain to divisions that are more global in nature. Lower levels move toward a stepwise refinement. Nested classifications must by necessity follow this plan. One must first specify the phylum before specifying the class in the animal kingdom. One must specify the country before specifying the city. Some hierarchies are completely nested. Other hierarchies may be crossed in the sense that lower levels share a number of alternatives in common. Linear menus are completely crossed since all levels have the same set of alternatives. Linear menus may be arranged in an arbitrary order but they may be effectively organized by scope. A news service may ask for region at the first level (e.g., world, national, and local); type of news at the second level (e.g., politics, business, science, arts, and society, etc.); and date at the third level. Since the attributes are crossed, the service could have date at the first level, type of news at the second, and region at the third.

Refinement. Objects may be classified as members of a linear sequence by successively subdividing the linear order. For example, the name Smith may be classified in an alphabetic list by first selecting Quigley-Sutton, then Silverman-Sutton, then Sloane-Smithers, and so on until single names are listed. In some cases this approach may be more efficient than typing in the name, depending upon the ratio of letters to type versus menus to select.

Subjective Importance to the User. The order of attributes may be programmed to agree with the order of importance to the user. Attributes along which objects seem to vary the most are presented first. For example, in classifying documents on one's disk, the user may first make a distinction between personal and business. Within personal, they may then be classified by addressee, and finally by date.

Formal Order. It is often the case in formal data collection that information is received in a prescribed temporal order and form. For example, in classifying claims for health insurance, the prescribed order corresponds to the printed claim form and takes advantage of the interaction mode of form file.

Natural Script. Finally, the order of attributes may follow a natural course of events or the process of some mental activity. In this way input follows the temporal progression of events and fits the expected protocol of interaction. For example, classification of automobile accidents may first specify location, direction, and speed of the moving vehicles; second, the point of impact; third, the amount of damage done; and finally, the determination of whose fault it was. By following an expected order of input the user anticipates each choice point and interprets it in light of a well-understood sequence.

3.9 SUMMARY

A variety of tasks may be performed using menu selection. In this chapter it was proposed that a task analysis be used to determine the functions to be implemented via menu selection. A theory of cognitive control was set forth in which system complexity/functionality was related to user/task need. It was shown how the system state diagram matrices should optimize user performance. A fourfold taxonomy of specific functions of menu selection was proposed listing pointing, command control, output, and input. Finally, four major types of systems were discussed which make use of menu selection, command menus, decision menus, information menus, and classification menus.