NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

DESIGNING THE USER INTERFACE:
CONSIDERING THE CONCEPT
OF COMPLEXITY

by

John B. Frank, Jr.

September 1991

Thesis Advisor: Kishore Sengupta

Approved for public release; distribution is unlimited

T259758

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) AS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
Designing the User Interface: Considering the Concept of Complexity

12. PERSONAL AUTHOR(S) John B. Frank, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	September 1991	139

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Interface Complexity; User Complexity; Cognitive Complexity; Task Complexity

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

The human-computer interface may be defined as the dialogue that allows communication between the human and the computer, the purpose of such dialogue being the accomplishment of some task. This thesis explored the relationship between task complexity, interface complexity, and user performance in the context of direct manipulation interfaces. Two different levels of task and interface complexity were introduced to subjects in two groups. Each group was presented with the identical task sets. There were three tasks sets, one a practice set, one a simple set requiring five inputs, and a complex task set requiring 24 inputs. The dependent variables measured were 1) task completion time, 2) number of errors committed, and 3) number jof help references needed. Results indicate that the complex interface took longer to learn, and more errors were made while learning. Results for the simple task set favored the simple interface as well, but once the subject learned the interface, the completion time was shorter and there were fewer errors made during the accomplishment of the complex task set on the complex interface. With an increase in task complexity, subjects using the complex interface showed an improvement in performance.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS REPORT ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Kishore Sengupta	22b. TELEPHONE (*Include Area code*) (408)646-3212	22c. OFFICE SYMBOL Code AS/SE

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

Designing the User Interface:
Considering the Concept
of Complexity

by

John B. Frank, Jr.
Commander, United States Navy
B.S., University of New Mexico

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

ABSTRACT

The human-computer interface may be defined as the dialogue that allows communication between the human and the computer, the purpose of such dialogue being the accomplishment of some task. This thesis explored the relationship between task complexity, interface complexity, and user performance in the context of direct manipulation interfaces. Two different levels of task and interface complexity were introduced to subjects in two groups. Each group was presented with identical task sets they were asked to accomplish. There were three task sets, one a practice set, one a simple set, and a complex task set. The dependent variables measured were 1) task completion time, 2) number of errors committed, and 3) number of help references needed. Results indicate that the complex interface took longer to learn, and more errors were made while learning. Results for the simple task set favored the simple interface as well, but once the subject learned the complex interface, the completion time was shorter and there were fewer errors during the accomplishment of the complex task set on the complex interface. With an increase in task complexity, subjects using the complex interface showed an improvement in performance.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

1. Man-Machine Interfaces 3

2. The Issue of Complexity 4

3. Interface and Task Complexity 5

a. Interface Complexity 6

b. Task Complexity 6

B. GOAL OF THIS STUDY 6

II. INTERFACE COMPLEXITY 8

A. DEFINING COMPLEXITY 9

1. Task Complexity 9

a. Task Complexity as a Psychological Experience 10

b. Task Complexity as an Interaction between the Task and the Person 10

c. Task Complexity as a Function of Objective Task Characteristics 10

d. Derivation of Task Complexity 11

e. A Framework for Objective Complexity 11

(1) Multiple Paths 12

(2) Multiple Outcomes 12

(3) Conflicting Interdependence Among Paths 13

(4) Uncertain or Probabilistic Linkages 13

(5) Other Associated Characteristics 14

f. Task Complexity as Opposed to Task Difficulty 14
2. Interface Complexity 15
a. Task Representation 15
b. Device Representation 17
c. Job-Task Representation 17
3. Problems with using Cognitive Complexity Models 19
4. Related Studies 20
B. INTERFACES 21
C. DIRECT MANIPULATION INTERFACES 21
D. HYPOTHESES 22
E. SUMMARY 23

III. METHODOLOGY 25
A. EXPERIMENTAL DESIGN 25
1. Independent Variables 26
2. Dependent Variables 26
B. SUBJECTS AND TASKS 27
1. Subjects 27
2. Task Sets 28
C. SETTING AND PROCEDURE 29
D. APPARATUS 31
1. Simple Direct Manipulation Interface 31
a. Practice Task Set Screen 37
b. Simple Task Set Screen 37
2. Complex Task Set Screen 37
3. Complex Direct Manipulation Interface 40
a. Practice Task Set Screens 52
b. Simple Task Set Screens 53
c. Complex Task Set Screen 55

E. DEPENDENT MEASURES 55
1. Primary Measures 55
2. Secondary Measures 56
3. Subject Data Files 56

IV. RESULTS 57
A. DATA ANALYSIS 57
B. EXPERIMENTAL RESULTS 57
1. Task Completion Time 62
2. Number of Errors 64
3. Help References 65
C. SUBJECT RESPONSES 69

V. DISCUSSION AND CONCLUSIONS 73
A. GOAL 73
B. HYPOTHESES 73
C. CONCLUSIONS OF STUDY 74
D. POSSIBILITIES FOR FURTHER RESEARCH 75

APPENDIX A - DEFINITIONS 76
APPENDIX B - SIMPLE DIRECT MANIPULATION INTERFACE 78
APPENDIX C - COMPLEX DIRECT MANIPULATION INTERFACE 100
APPENDIX D - SAMPLE USER LOG 124
APPENDIX E - CODE MEANINGS FOR SIMPLE INTERFACE USER LOG . 125
LIST OF REFERENCES 126
BIBLIOGRAPHY 128
INITIAL DISTRIBUTION LIST 132

I. INTRODUCTION

A. BACKGROUND

This study begins with the assumption by the author that people can be grouped into three broad categories based on their perception of the computer and how it affects them.

The first category would include those who are fascinated with the technology and the challenges it presents. They enjoy exploring the limits of the machine and devising unique ways to push the computer to those limits. Once they have reached the limit, they develop newer and more capable hardware and software. They consider the computer as something more than an aid to enhance productivity in the business world or a tool to make possible complex operations in mathematics and engineering, which in turn furthers research and development. Those who may be in this category are computer scientists, creative programmers, computer hobbyists, and those who, in their spare time, write computer programs that are placed in the public domain for anyone to copy and use for free or for a nominal fee (shareware).

Another category includes those who appreciate the capabilities of the machine, but have no desire to understand anything further than how to operate the machine in such a manner as to gain some advantage that would not be feasible without the computer. These individuals see the computer as a sophisticated tool. They may be architects, engineers, and draftsmen who use computer aided design (CAD) applications; scientists and engineers who use the computing power of the machines; accountants, businessmen, secretaries, desktop publishers, medical doctors, and others. The list of users is as long as the list of applications that have been developed in response to the demands of professionals, academicians, businesses, and the general public.

The final category includes those who have had no experience or very limited experience with a computer. Within this category is an important group who could stand to benefit by becoming computer users, but who have not done so. The reason some have not become users is that they are intimidated by the computer and will go to great lengths

to avoid any type of interaction with the machine. This attitude persists even with the realization that there is a potential for personal loss because others are gaining an advantage by embracing the technology and are using the computer to their benefit. The individuals in this group are generally those who did not see a need to become computer literate in the past, when the introduction of the personal computer put the power of the machine within easy reach of the average person. Now that the rapidly declining cost of these devices has encouraged many people to purchase one for personal use and they are an integral part of the typical business office, these people find themselves in an awkward position. They are becoming part of a shrinking population who are unable to perform even the most fundamental tasks with the computer. Most entry level office workers either have a basic working knowledge or acquire that knowledge quickly on the job. A growing number have a really sophisticated understanding of computing.

This third category includes a large number of people. Many of them will not suffer any personal disadvantage by not being introduced to the computer. They have no need now, and probably never will. The others, however, could benefit by learning to use the technology. Some examples are schoolteachers who could impart the knowledge to their students, or set an example in learning; students who are finding more evidence of the prominence of computers in every day life as they advance in their education; individuals who are entering the job market after raising families, but find that ignorance in the areas of word processing, electronic spreadsheets, electronic accounting, computer graphics, or database management systems is keeping them from being competitive.

Informal interviews with a number of people who are in this category have suggested the possibility that many people are reluctant to learn these new skills because there is a degree of intimidation associated with interacting with a computer, and that is preventing them from learning the very skills necessary to help them in education, make them attractive to an employer, or enhance productivity. There is an active research field exploring the anxiety factor in human-computer interaction.

The intimidation may be a result of a number of reasons, but one of these reasons could be that a number of these individuals are uncomfortable with the man-machine

interface, or the interaction between themselves and the computer. If the people in the third category could be introduced to the application of choice without having to suffer through the intricacies of learning to manipulate the computer interface, in other words to have the interface invisible or transparent to the user, leaving only intuitive operations to perform to complete the task, that intimidation would decrease to a level where dealing with it would be much easier.

Thus it is the job of the designers of interfaces to make their products so that they are attractive to the person who is not inclined to learn a new skill through great sacrifice. A term which is getting much use in recent years is "user friendly," which is a good sign of the direction which the computer using public is forcing the designers of commercial software application programs to take.

1. Man-Machine Interfaces

The first automobiles were fairly complex to operate. Compared to those early cars, today's feature rich automobiles are much more complicated, but much simpler to operate. The monitor for a personal computer, like a television, is a more complicated engineering project than the computer itself, but the television is much simpler to operate. The interface is the system whereby the input is communicated and the feedback is received. The person inputs requirements and the machine feeds back either the desired response, a need for more information, a response based on poor information or an error message.

In an automobile, the ignition starts the engine, the transmission determines direction of travel and the engine RPM and gearing determine the vehicle speed. The operator is only concerned with turning the key in the proper direction the required distance, placing the gearshift in the proper position and depressing the accelerator the required amount. How each of the functions is accomplished is of little concern to the average driver.

A television set has a power switch to turn it on and a channel selector to allow programming. The average operator has to know little beyond the location of these controls and how to operate them.

These familiar objects, and a host of other technologically advanced but easy to operate devices, are in great supply today. The average person has not achieved that degree of familiarity with the computer. It is not always a simple matter for a novice user to translate requirements from a normal human thought process to the proper form for input to a computer and see the result in the desired output. The interface may be a reason that this has not happened. Rassmussen states that a good user interface is one of the most important components of an information system. The value of the system directly depends on whether or not it has provided a useful user interface (Rasmussen and Zunde, 1987: p. 152)

Since the wide range of capabilities that the computer brings are so impressive, it is in our best interest to use these devices if our goal is to lessen our individual work load or to increase productivity. Even the most powerful computer developed is of no benefit if it is unused. It therefore becomes a priority issue for those designing the computer interfaces to insure that the interface is as useable as possible, which generally means that it as transparent, or as invisible to the user, as possible. There are two suggested ways to look at this idea of interface transparency. One of these is the degree of difficulty required to learn the operation of the interface. The other way is to what degree knowledge of an interface already learned can be transferred to permit operation of a new interface. An important question is what is the impact of a complex interface on the user, or, more fundamentally, what makes an interface complex.

2. The Issue of Complexity

The subject of complexity was examined by Kieras and Polson in an article written in 1983. The approach was to illustrate how a prediction could be made of the effort required to learn a new interface to the point that the user could show productivity. The application used in the study was a commercially available word processing package. They proposed a method of analyzing the concept. The broad approach is to look at the complexity of a device from the point of view of the user. This means determining the degree of knowledge possessed by the user on how to use the device and how the device behaves. According to the authors, the device can be described in terms that can be

related to the knowledge required to operate it. A formal description system was proposed for representing interactive devices and how the relationship between the device and the knowledge required to operate the device can be described. Two representation systems were presented. The first represents the device, and the second represents the complexity of the user job or task.

User interfaces, the hardware and the software, are the devices with which this thesis will be concerned.

3. Interface and Task Complexity

To begin an analysis of human computer interaction, Card, Newell and Moran presented the model of the Human Information-Processor. This was an attempt to describe the psychological knowledge about human performance as it is relevant to human computer interaction. This model is called the Model Human Processor, and can be described by three interacting subsystems: 1) *the perceptual system,* 2) *the motor system,* and 3) *the cognitive system.* The perceptual system receives and holds sensory information while it is being coded. The cognitive system receives the coded information and uses information in memory to make decisions as to how to respond. The motor system carries out the response (Card, Moran, and Newell, 1983).

Concentrating on the cognitive system, a description was suggested based on a model called the GOMS model (Card, Moran, and Newell, 1980, 1983). This describes the user's understanding of the task's goals, operations, methods and selection rules.

In the previously mentioned work by Kieras and Polson, an explanation was given as they applied to the GOMS model a model called the production system, which was proposed in 1972 as a tool for building formal models used in theories of problem solving (Newell and Simon, 1972). In this system, mental processes are represented as specific responses to particular stimuli. A production system contains a set of rules and a "working memory." The working memory contains the current goals of the system, and information about and inputs to the environment. A direct relationship can be drawn between the GOMS model and the production system. This relationship will be examined in Chapter II.

a. Interface Complexity

The complexity of the interface is determined by how the interface is represented by the user. There are four categories of information that the user has in his knowledge of the interface. He knows what tasks the interface can satisfy. He knows the operating layout of the interface. He knows how the interface will behave in response to actions on his part. He knows something of the internal structure or functionality of the interface (Kieras and Polson, 1985: p. 367). The complexity of the interface derives from the complexity of the knowledge required to operate the device.

b. Task Complexity

Task complexity is defined by how the user represents the task, which is also a major component in the knowledge required to operate the interface. This representation is described by the GOMS model introduced earlier. A further distinction is made between the interface-dependent and interface-independent knowledge in the task representation.

B. GOAL OF THIS STUDY

Cognitive complexity issues have been applied to man-machine interfaces, or user interfaces as they will be predominately called in this study, with interesting results. Different types of interfaces have been studied as well as varying degrees of complexity of task sets.

The goal of this study is to measure the relationship between task complexity and interface complexity on user performance. Reinhard showed that direct manipulation interfaces have an advantage over command language interfaces above a certain degree of task complexity. In her research, two degrees of task complexity were used in each of the interfaces, one simple and one complex (Reinhard, 1991).

This study will examine the twin issues of task complexity and user interface complexity in the context of direct manipulation interfaces. The task sets are on two

levels of complexity. One is simple and the other complex. The tasks are those normally associated with operating system functions, such as directory and file manipulation and editing.

II. INTERFACE COMPLEXITY

The purpose of this chapter is to establish a point of reference from which to describe the study. Complexity will be defined as it relates to the task and to the interface, and then the issue of cognitive complexity will be briefly described. Since this study follows the approach of examining complexity from the user's perspective, a bit of background relating to the cognitive complexity model (a model developed by Kieras and Polson in their extensive writings in the field), is provided. Finally a number of hypotheses regarding the effect of interface complexity on productivity will be presented.

The key notion in defining the human-computer interface is that the computer and the human are engaged in some sort of exchange, the purpose of which is to accomplish some task (Suchman, 1987: pp. 13-19). This exchange is actually a dialogue in which symbols flow to allow communication. During the course of this dialogue, each entity can interrupt, query and correct the communication. Everything involved in this exchange makes up the interface. This includes the physical devices, such as the keyboard or other input devices and display hardware, as well as the software that controls the dialogue (Card et al., 1983: p. 4).

From this concept, it is clear that interface design plays a critical role in establishing the effectiveness of that interface to promote a productive dialogue between user and computer.

The term "cognitive complexity" was used by Kieras and Polson (1985) in their proposal for a formal approach to analyzing the complexity of a device. They began their study using the term "user complexity" because the complexity they were analyzing was from the users' point of view, but later changed the terminology. Using Rasmussen's definition of complexity in the design of interfaces is the concept this study will explore.

A. DEFINING COMPLEXITY

Rasmussen defined complexity as the number of goals and processes the operator of a system must control. The number of the processes and goals as well as the means available to control them determine complexity (Rasmussen and Zunde, 1987: p. 24). Any research on the topic of interface complexity will quickly reveal that complexity in this case is defined from the users' point of view, and that the areas to be analyzed are the task itself and the device. The device in this study will be the interface of interest. In the following sections these two areas, task and interface complexity, will be discussed, and relevant previous studies on the subject will be described.

1. Task Complexity

Task complexity has been examined in at least three areas of research: the information processing and decision making literature, the task and job design literature, and in the goal setting research literature. These areas have a common denominator that is be meaningful to the current study. Complexity is seen as: (a) primarily a psychological experience, (b) an interaction between task and person characteristics, and (c) a function of objective task characteristics (Campbell, 1988: p. 40). The concept of complexity as a function of objective task characteristics will be explored in more detail later.

Wood proposed a general model of tasks in which all tasks contained three components; products, actions, and information cues. In the model these three components are used to derive dimensions to analyze the complexity of the task. These analytical dimensions are: component complexity, coordinative complexity and dynamic complexity (Wood, 1986: p. 60). This approach is theoretical and is offered as an alternative to the empirical approach in which task characteristics are derived from individual perceptions of the task.

This theoretical model is similar to the objective task characteristics discussed later in this chapter. Wood proposes is that task complexity can be measured if the task inputs and outputs are known. The inputs are the acts necessary to accomplish the task

and the task information cues that are available to the user. These inputs are processed and a product is formed. This product is the output (Wood, 1986: pp. 66-74).

a. Task Complexity as a Psychological Experience

The psychological dimensions of the task are generally thought of as having to do with factors such as perception by the user of the task significance and the task identity. The emphasis is generally on the reactions of the individuals to the tasks rather than on the characteristics of the task itself. It is a very subjective area, but the objective task characteristics are not entirely ignored. The task characteristics must be considered to some degree to bring about the psychological state (Campbell, 1988: p. 41).

b. Task Complexity as an Interaction between the Task and the Person

Looking at the task as a person-task interaction is more intuitively plausible. Tasks are easily examined in terms of how complex they are relative to how the user perceives his or her capabilities of performing the task. An important distinction to make here is the difference between task complexity and task difficulty. More will be said about this distinction later. Looking at task complexity in this manner, it can be seen that the degree of complexity can vary with the skills and the insight of the individual accomplishing the task. This implies that the complexity of a task cannot be determined without considering the individual's short term memory, span of attention, computational efficiency, and other capabilities as they apply to the particular task, and how they are affected by the task representation (Campbell,1988: p. 42).

c. Task Complexity as a Function of Objective Task Characteristics

In this attempt at defining task complexity, researchers have suggested examining the task according to the individual's reactions to the task as it impacts on the five senses. Reaction is measured by the magnitude and variation of the stimulation and the number of senses affected. Other objective task qualities that contribute to complexity are unknown or uncertain alternative operations to accomplish the task, unknown consequences of actions taken to accomplish the task, inexact or unknown means-ends connections, and the number of sub-tasks. Also to be considered is the path-goal

multiplicity, or the number of ways to accomplish the same goal. This concept relates to the means-ends connections. This can be further defined as (1) there appears to be several ways to achieve the goal, but only one way actually accomplishes it, or (2) there are several ways to do the job, and it is up to the user to determine the best or most efficient method. Complexity would appear in the requirement to decide which path represents the optimal one. Complexity varies with the number of interrelated and conflicting elements and the number of rules to satisfy. Complex tasks place high cognitive demands on the individual. These demands are a result of the task itself, and in no way reflect on the capabilities of the individual (Campbell, 1988: p. 42).

d. Derivation of Task Complexity

One way to define task complexity is as a function of the quantity of alternatives offered to perform the task and the number of attributes with which each alternative is compared. Another approach is to evaluate three basic properties: (1) The number of dimensions of information requiring the user's consideration, or the information load, (2) the number of alternatives associated with each dimension of information, and (3) the rate of change of the information received, or the uncertainty of the information. As each one of these properties increases quantitatively, so does the complexity of the task.

This approach equates complexity with the amount of information associated with the task, and the stability or certainty of that information. The number of alternatives that are associated with the information determines the diversity of the information input (Campbell, 1988: p. 43).

e. A Framework for Objective Complexity

As Campbell suggests "...any objective task characteristic that implies an increase in information load, information diversity, or rate of information change can be considered a contribution to complexity". In his framework, four task characteristics meet this requirement: (1) multiple paths to arrive at the desired end state, (2) the possibility of multiple end states, (3) conflicting interdependence between multiple paths to multiple

end states and (4) uncertainty of the links between multiple paths and end states (Campbell, 1988: p.43).

Complex tasks can be classified by using these attributes, determining the degree to which the individual attribute is incorporated into the task and determining the number of individual attributes that the task contains (Campbell, 1988: p. 46).

(1) Multiple Paths. The number of paths available to achieve the desired outcome directly determines the amount of information input. The higher the number of alternative paths required to be followed, the greater the information load, and therefore, the greater the degree of complexity. This does not apply to the principle of redundancy. If all the paths involved result in the same end, there is redundancy and the task may actually be simpler because the individual has an option to use the path that is easiest to learn. The condition of multiple paths increases complexity only when it appears that there are several paths and only one leads to the desired goal. Or, there are a number of paths, but the optimum or most efficient one must be chosen. In these two cases, complexity grows with the number of paths available (Campbell, 1988: p. 43).

This notion parallels theoretical model of component complexity, which he describes as a direct function of the number of individual, distinct actions that are required to be accomplished and the number of information cues that must be processed during the completion of a task. As the number of actions increases, the knowledge of the individual must also increase because there are more activities to perform (Wood, 1986: p. 66).

(2) Multiple Outcomes. The more desired outcomes of a task, the higher the complexity of the task. Each outcome of the task should be thought of as a separate task dimension that requires action. Each dimension can be thought of as a separate information processing operation, and as the number of dimensions increases, the number of information processing operations increases. Campbell notes one exception to this general statement. "If the desired outcomes are positively related, the degree of complexity is reduced. The positive relationship builds in redundancy." (Campbell, 1988: p. 44).

In Wood's theoretical model, his concept of coordinative complexity parallels both this idea of multiple outcomes as well as the next item for discussion -- the conflicting interdependence among paths. Coordinative complexity refers to the relationships between the inputs and the products of the task. If the product is a simple linear function of the inputs, the coordinative complexity is low and the task is a simple one. If there is a nonlinear relationship between the product and the information cues or actions, knowledge of the turning points in the function is required to successfully accomplish the task. The number of turning points in the function will describe the relationship between the inputs and the products. Generally, the higher the value, the more complex the task (Wood, 1986: p. 70).

(3) Conflicting Interdependence Among Paths. An opposing idea is that complexity can occur because of negative relationships among the desired outcomes. In other words, if achieving one outcome conflicts with achieving another outcome, the degree of complexity in the task will increase. One illustration of this conflict would be the question of quality or quantity, assuming the two goals are incompatible. The decision the individual must make is either a compromise or the choice eliminates one of the desired outcomes.

(4) Uncertain or Probabilistic Linkages. The operation of processing the information received will be greater if the individual cannot establish the relationship between the path and the desired outcome. If the alternatives include probabilistic linkages, meaning that potential paths cannot be quickly or easily eliminated, or if there is a degree of diversity, meaning that different action-outcome pairs must be evaluated, the information load will be greater and the complexity will increase. Uncertainty will increase the number of potential paths as well, thus increasing complexity. If there is uncertainty as to the number of potential paths, meaning that there is not a firm, known bound on the number of paths, then consideration must be given to the possibility of another potential path, which could be better, and the information pool becomes greater and the complexity increases.

The concept of dynamic complexity of the theoretical model proposed by Wood is similar to this ideal of uncertain or probabilistic linkages. Dynamic complexity is caused by changes in the environment that change the relationships between task inputs and products. These changes in the relationships can change the knowledge and skills required to do the task (Wood, 1986: p. 71).

(5) Other Associated Characteristics. Besides these four basic attributes, there are other characteristics that can be associated with task complexity. Such characteristics as lack of structure, ambiguity and difficulty do not have a straightforward relationship to the objective classification of task complexity. Complex tasks often have a poor structure, are vague and are difficult, but these characteristics derive from the more fundamental attributes described previously. A task having multiple, loosely linked paths with several conflicting outcomes is going to be poorly structured, ambiguous and difficult. These associated characteristics result from a complex task possessing the fundamental attributes, but the reverse is not always true. In other words a task that is poorly structured, vague or difficult may be uncomplicated and straightforward, but a communication failure may be responsible for the difficulty. An external factor that is not a part of the task itself has made the task to be experienced as a complex one. This complexity results from the lack of clarity caused by the poor communications (Campbell, 1988: p. 45).

f. Task Complexity as Opposed to Task Difficulty

Campbell declares "Complex tasks are, by their nature, difficult. Thus, sometimes the two notions can be used interchangeably, but not always." This point was illustrated by the example of digging a foundation and planting a flower. Neither task could be described as complex, but digging the foundation is certainly more difficult than planting a flower. Tasks can be quite difficult, meaning that they require much effort, but they are not necessarily complex. On the other hand, some tasks are difficult because they *are* complex.

Another important concept is the implication that task difficulty brings in the interaction of the individual and the task. A task that is difficult for one person

may not be difficult for another, even though the objective characteristics or attributes described earlier are identical.

2. Interface Complexity

Interface complexity is the complexity of the system from the user's point of view. This is the user's knowledge of the device, how to use the device and the behavior of the device, and of the task. As mentioned earlier in Rasmussen's definition of complexity, the means available to control the goals and processes are a factor in determining complexity. The complexity of the task has been briefly discussed earlier. Attention will be turned to the user interface, which will require the complexity of the task to be considered in determining the interface complexity.

Kieras and Polson proposed that the complexity of the device depends on the knowledge required to use it. In addition, it is also a function of the difficulty of acquiring the requisite knowledge to successfully operate the device by a new user. To determine the knowledge required, it is necessary to examine two representations. These are the device representation and the task representation that the user has developed. The task representation is the user's knowledge of how the task is to be accomplished using the particular device, and the device representation is the knowledge possessed by the user of the device itself. Therefore it has been theorized that the device complexity depends on the complexity of the user's task representation, or how much learning, memory and information processing is required by the task, the number of device-dependent functions which are independent of the initial task representation and the difficulty of obtaining operational knowledge of the device (Kieras and Polson, 1985: p 365-366).

a. Task Representation

To describe the task representation, an understanding of one of the more accepted models is necessary. Card, Moran and Newell used the GOMS Model to define and predict human computer interaction. The GOMS Model assumes four components of the human cognitive structure, and the model is composed of a user's understanding of *goals, operations or operators, methods, and selection rules*. Goals are defined by decomposing the task into the elementary parts that can be accomplished on the device.

They are hierarchical and sequential and altogether define the *goal structure*, which is considered the plan for carrying out the complete task (Card, Moran and Newell, 1983: p. 139-144). Goals can also be the representation of the user's intention to perform a task (Bovair, Kieras and Polson, 1990: p. 8).

Operations are the mental visualizations of the elementary functions the device can perform as well as other cognitive functions that occur during the execution of the task (Kieras and Polson, 1985: p. 366).

Methods are procedures for satisfying specific goals. This is defined anywhere from down to the most elemental form of a single keystroke to as complex as the entire hierarchy of goals with the operations required to satisfy them (Kieras and Polson, 1985: p. 366). Methods can be thought of as a sequence of operators performed to achieve some goal (Bovair, Kieras and Polson, 1990: p. 8).

Selection rules specify what methods are to be used to accomplish a specific goal or subgoal. The selection rules are used when there are several methods that can be used to accomplish a specific goal and each has different characteristics. The method that is most appropriate in the given context is selected (Kieras and Polson, 1985: p. 366).

The framework of the GOMS Model is applied to the environment of the user as it applies to the task. The job-task environment[1], or the user's understanding of the job situation and the tasks that appear in the job situation is used to create the job-task representation. The job related portions of the representation use only two of the GOMS components, the *job goals* and the *selection rules* (Kieras and Polson, 1985: p. 367).

Here a distinction needs to be made between the *device-dependent* and the *device-independent* knowledge in the task representation. Device-dependent knowledge consists of the knowledge of functions needed to accomplish a certain task only as they apply to the device being used. Device-independent knowledge consists of the knowledge of the task itself, regardless of the device used to accomplish it. Creating

[1] See Appendix A for definitions

a manuscript requires certain knowledge of formatting and rules of grammar and such, regardless of whether it is to be written on a typewriter or a word processor. However, there is other knowledge required to accomplish pertinent tasks on the device of choice. This distinction is important and is central to this thesis. If a new user can apply *device-independent* knowledge, the system will be easier to learn. If there is a great degree of *device-dependent* knowledge, the device will be relatively difficult to learn (Kieras and Polson, 1985: p. 367).

b. Device Representation

In a paper published in 1982, Kieras and Polson assumed four categories of information in the user's device knowledge base. These were: (1) *task-relevant* knowledge, (2) *device layout knowledge,* (3)*device behavior knowledge*, and (4) *how-it-works knowledge.* Task-relevant knowledge is the counterpart to the user's task representation. This is the information the user has of the goals the device can be used to satisfy, the operations that can be performed on the device, and the device operating procedures. Device layout knowledge is information the user has of the physical layout of the device. This would include the location of the controls and indicators and the display format. Device behavior knowledge is the information as to how the device responds to control inputs. How-it-works knowledge is the user's understanding of the fundamentals of the device operation, or *how* it does what it does.

The complexity of the device can be shown to consist of the knowledge representation required to operate it and depends on the complexity of the user's task representation, the number of device-dependent functions and the degree of difficulty required for the user to acquire how-it-works knowledge (Kieras and Polson, 1985: p. 367).

c. Job-Task Representation

The model chosen in this study to represent task complexity is the *cognitive complexity* model proposed by Kieras and Polson. This model is a formalized and quantified GOMS model. In developing this approach, the structure of the device was separated from the structure of the user's knowledge and the two were treated

differently. The production system concept was used to develop this model, which is an old concept in psychology with a new application by researchers in the field of man-machine interactions. The concept is that behavior or mental processes can be represented as a set of actions made in response to a stimulus. The use of the production system makes it easy to formulate the difference between declarative knowledge and procedural knowledge, or the knowledge of facts and the knowledge of how to do something. The system is composed of a collection of rules, a working memory and an interpreter. The working memory contains representations of current goals, other information concerning the status of current and past actions and inputs from the environment. A production rule is a condition-action pair, shown as:

IF (*condition*) THEN (*action*)

The *condition* of a production rule is a statement about the contents of the working memory, such as if certain specified goals are present or what are the environmental inputs. If the condition is true then the rule is said to *fire* and the action component is executed. A set of these production rules is a program, and the process of executing the program needs to be specified. The interpreter operates by alternating between the *recognize* and *act* phases. During the recognize phase, the interpreter matches the conditions of all rules against the contents of working memory. During the act phase, all rules that match will fire, and the interpreter will execute their actions (Bovair, Kieras and Polson, 1990: p. 7-9).

The relationship of the GOMS model and the production system concept used in this approach is a direct one. Goals are directly represented for job-task knowledge. They appear as the conditions of the production rules. They are manipulated in the production actions. Methods appear as the sequence of production rules, the first one of which is triggered by the assertion of the goal of the method. Selection rules are the production rules that control the execution of the methods. Operations or operators exist throughout the entire system as either elementary actions or environment-testing conditions.

The point of constructing these models of the job-task representation is to achieve a means of measuring and evaluating complexity, and allow a prediction of the degree of difficulty that will be experienced by a new user in learning a new device. This will allow a precise prediction to be made as to how long it will take a new user to acquire sufficient knowledge to be able to use the device.

3. Problems with using Cognitive Complexity Models

There are some significant problems in using cognitive complexity models that have had to be addressed (Kieras, 1988). The first is the recognized difficulty of constructing production rule simulation models. Another is the difficulty of doing, in a standardized and reliable way, the detailed task analysis required to construct the representation of the procedural knowledge that the user must have in order to operate the system.

An approach is production rule formalism, which is the standard and current theoretical idea for the representation of procedural knowledge. The problem with this is that writing production rules is an arduous task, done in assembler language (Kieras, 1988).

The GOMS model is a higher level language but contains no detailed explanation as to how notation works. In addition, it is clumsy to use and provides a weak connection to underlying cognitive theory because there is no calculational base for making predictions of learning time from an explicit GOMS model (Kieras, 1988).

Task analysis is a very difficult part of using the cognitive complexity model because first there must be formulated a detailed and specific description of the procedures that the user must know in order to use the system with which the task is to be accomplished. This constitutes a GOMS model for the system. This can be a lengthy and difficult task, more suited to the cognitive psychologists than to the people who enlist the model in practical design methodology (Kieras, 1988).

Another problem is whether or not such analyses, which amount to the construction of psychological theories of how users represent tasks, can be done reliably and routinely by the kind of individuals who usually design computer systems.

Employing the model calls for intuition in making judgement calls and the routine description of the methods entailed by design (Kieras, 1988)

The Natural GOMS Language developed by Kieras addresses these problems by allowing the creation of procedural documentation which can be checked against the methods in the GOMS model. It affords the description or creation of a model of the knowledge that a user must have in order to carry out tasks on a device or system. It allows a representation of the operational knowledge required by a system in order to get the intended tasks accomplished (Kieras, 1988).

4. Related Studies

Karat, Fowler and Gravelle used the model proposed by Kieras and Polson in an attempt to examine the learning and performance differences between a command language and a direct manipulation interface. They exposed computer novices to computer file management functions and taught them to carry out a series of tasks on one of the two interfaces. The results revealed a large advantage in performance of the direct manipulation system over the command language interface. In this experiment the inability to predict error data discussed earlier was a basic problem with the formal model (Karat, Fowler and Gravelle, 1987: p. 489).

Reinhard (1991) conducted a similar experiment with the same basic results. The experiment did not show an advantage of one interface over the other in the accomplishment of simple tasks, but the direct manipulation interface showed an advantage over the command line interface when the tasks became more complex. The variables measured in the experiments were length of time required to accomplish the tasks, the number of errors committed in the course of accomplishing the task, and the number of times the on line help functions were accessed (Reinhard, 1991).

Margono and Shneiderman conducted an experiment which compared file manipulation operations done with an Apple Macintosh, which uses a direct manipulation interface, with file manipulation operations done on an IBM PC using MS-DOS, which is a command line interface. The subjects were a group of computer novices who had undergone a brief training period on the interface that they would be using. The

experiment showed that those using the direct manipulation interface accomplished file manipulations more rapidly, with fewer errors and greater satisfaction (Margono and Shneiderman, 1987: p. 154).

B. INTERFACES

More and more attention has been given to user interfaces in recent years. Part of the risks to be identified and managed in software development in such a competitive arena like commercial application software development can be directly attributed to whether the user community accepts the interface. Three basic types of interfaces have been developed and are generally accepted by a substantial number of proponents. These are the command line interface, such as operating systems like Unix and DOS; the menu type that is prevalent in popular commercial applications like the database management systems, spreadsheets and others; and the direct manipulation interfaces which have become popular with the "graphical user interfaces" brought into favor recently with the success of Microsoft Windows and similar interfaces. Most of the more popular applications employ a combination of these two, such as a menu type with a command line feature, or a direct manipulation with a menu feature.

Previous studies noted here have concluded that the direct manipulation interface is more productive and efficient when accomplishing a complex task set. This study accepts that conclusion and and moving from that conclusion will explore whether the complexity of the interface itself can be a factor in productivity. Only the direct manipulation type interface will be used in the experimentation.

C. DIRECT MANIPULATION INTERFACES

As Reinhard summarized, most people visualize in their minds tasks that need to be performed. Individuals may be able to more easily understand, learn and commit to memory the tasks and the steps necessary to accomplish those tasks when they are able to visualize the objects and actions. In a direct manipulation interface, the design objective is to achieve a visual representation of the objects and actions that match the way people think about the task.

This type of interface allows the individual to control action in the system by direct action on the objects represented pictorially in the interface rather than using procedural language to command the action. Feedback from the system is an integral part of the interface, providing instantaneous feedback to the user's actions. This feedback is then processed by the user's cognitive system (Reinhard, 1991: p. 21).

The pictorial representations of the objects and actions are called icons. When some action is directed at the icon with a pointing device, such as a mouse, an action is performed by the system that is reflected in the user's visualization of the task or operation. It seems to be easier to memorize and learn when the individual can visualize the task. Novices do better when they are able to associate actions and objects with pictorials, such as recognizing and selecting an icon, rather than having to memorize procedural language. Using procedural language requires the user to input commands in exact syntax (Reinhard, 1991: p. 22). The use of icons can reduce the complexity of an interface, making it easier to learn and easier to use.

The disadvantage of the direct manipulation interface is the number of physical operations that must be performed. The user must move his hand from the position on the keyboard, position his hand correctly on the mouse, move the mouse so that the desired icon has been identified, perform the mouse action that selects that icon, then return the hand to the correct position on the keyboard. This series of physical actions can inject irritating delays for the more expert users who are able to use procedural language to efficiently accomplish the task, never needing to have their hands leave the keyboard. The less experienced users don't notice the delay, since they are actually saving time by not having to learn the syntax of the command line procedural language.

D. HYPOTHESES

The hypotheses tested in this experiment were built around the three variables studied. The variables of time for task completion, number of errors committed and the number of times the help screens were consulted were applied quantitatively to test the hypotheses. The hypotheses are offered for the alternative case.

1.) Hypothesis 1: Completion Time

H_{1A} The time required to learn the complex interface will be greater than the time required to learn the simple direct manipulation interface.

H_{1B} The time to complete the simple task set will be similar for both interfaces.

H_{1C} The time required to complete the complex task set with the complex interface will be less than that required to complete the simple interface.

2.) Hypothesis 2: Number of Errors

H_{2A} The number of errors that occur during the practice set will be greater for the complex interface than for the simple interface.

H_{2B} The number of errors committed during the simple task set will be similar for both interfaces

H_{2C} The number of errors committed during the complex task set will be greater for the simple interface than for the complex interface.

3.) Hypothesis 3: Number of Help References

H_{3A} The number of times the help screen is accessed will be greater during the practice session for the complex interface than for the simple one.

H_{3B} The number of times the help screen is referenced during the simple task set will be similar for both interfaces.

H_{3C} The number of times the help screen is referenced during the complex task set will be greater for the simple task set than for the complex one.

E. SUMMARY

The complexity of the user interface is an integration of two components; the complexity of the task and the complexity of the user interface. The cognitive complexity model developed by Polson and Kieras uses the GOMS model presented by Card, Moran and Newell and applies to it the production system. The cognitive complexity approach affords a method whereby the ease with which a user can learn a new device and proceed

with the accomplishment of a task can be predicted. Earlier studies have determined that the direct manipulation interface is less complex than the command line interface, reducing by a measurable degree the complexity of the user interface as the complexity of the individual task increases, when both interfaces are compared.

III. METHODOLOGY

In studying the effect of interface complexity, this study will concentrate on the direct manipulation interface, introducing two levels of complexity in the same type of interface. The simple interface is the interface developed for the Reinhard study (Reinhard, 1991: pp 39-46). A more complex interface was used in comparison. A commercial graphical user interface for MS-DOS was selected because of the extensive functionality offered and the redundant or alternative paths available to accomplish given tasks.

Two levels of interface complexity and two levels of task set complexity were used. The tasks were restricted to those normally associated with the basic computer operating system. These were file and directory manipulation tasks, presented in varying degrees of difficulty. The task sets chosen were almost identical to those selected in Reinhard's study, in order to compare to an existing data base.

Fifteen subjects participated in the study for one interface and fourteen for the other, totaling 29 subjects. The subjects were a cross section of military students attending graduate classes in information systems and telecommunications.

A. EXPERIMENTAL DESIGN

This study was designed as a follow-on to Reinhard's work in 1991. The results of the original experiment showed that when the complexity of the task increased, the direct manipulation interface was more effective than the command line interface. The same design as the original experiment was used for the sake of consistency and to prevent adding new variables. As illustrated in Table 1, the type of interface (complex or simple DMI) was the between-subjects factor, differing from the previous work in that the same type of interface was used with two levels of complexity. The type of task was the within-subjects factor and was the same as the previous study.

TABLE I
EXPERIMENTAL DESIGN

	Interface Type	Type of Task	
Between Subjects	Simple DMI	Simple Task	Complex Task
	Complex DMI	Simple Task	Complex Task

Within Subjects

1. Independent Variables

The independent variables for this study were (1) the complexity of the interface and (2) the complexity of the task. There were two direct manipulation interfaces used, with different levels of complexity. The simple interface was the same one used in the experiment Reinhard conducted (Reinhard, 1991). The complex interface was derived from a popular commercial of direct manipulation interface. Both interfaces were designed in the MS-DOS environment.

Task complexity was calculated the same as it was in the original experiment, once again to keep the conditions as similar as possible. The complexity was calculated by determining the number of inputs and outputs required for each task. The simple task set had an average of five inputs and outputs. The complex task set had an average of 24 inputs and outputs. Therefore, the complex task set had an average of 80% more inputs and outputs than the simple task set.

2. Dependent Variables

Data were collected for each user as each task set was performed. A datafile was created automatically for the simple interface, logging the total completion time, an operation code, and a description of the operation. Any errors that occurred were coded to facilitate analysis. For the complex interface, manual data collection was required. Time to complete the task, number of times the online help was accessed, and the number of errors were logged for each task set.

Two separate task completion times were calculated. The total task time was when the subject started the task set and ended when the last task was completed. Time

to accomplish individual tasks in the task set was also recorded to analyze trends. The practice session was recorded in the same manner, but some subjects ran through the practice set more than one time before they felt comfortable enough to proceed.

An error was considered any action that produced any result other than the one intended. Any error that was recorded automatically in the datafile for each user on the simple interface was counted as well as others that were discovered during a manual review of the individual user files. The automatic data logging feature made no determination as to correctness of the task as it applied to the task, only to the legality of the action performed. It was therefore necessary to review the user log to determine if the actions were correct to accomplish the assigned tasks.

The number of times that the online help screens were accessed was automatically recorded by the simple interface and manually recorded in the complex interface. The number of times a subject had to consult the help screen was considered an indicator of the difficulty being experienced in remembering how to use the interface after reading the instructions and completing a practice round.

B. SUBJECTS AND TASKS

1. Subjects

Twenty nine subjects were recruited from the Naval Postgraduate School, Monterey, California. All the subjects were active duty United States military officers enrolled in master's degree programs in either computer science, computer systems management, or telecommunications. Five of the subjects were female and the remaining 24 were male. The age range was from 26 to 42 years, with the average age being 33.55 years. None of the subjects had any experience on the interface they used, and the selection was completely random. The level of computer knowledge and experience varied from very little to expert.

2. Task Sets

The actual task sets were the same for each interface. For the simple task set, users were required to manipulate 17 files in 2 directories. The task set consisted of the following:

1. Create a subdirectory of \ called *plots*.
2. Create a file called *twoplots.drs* in the *plots* directory.
3. Delete the file called *project.bak* in the *project* directory.
4. Copy the *plot.drs* file in the \ directory to *plot.bak* in the \ directory.
5. Rename the file called *twoplots.drs* in the *plots* directory to *twoplots.bak*.

The second task set was made more complex by expanding the directory system and by constructing the tasks as a combination of operations. A total of 113 files in 17 directories were used as the directory system for the complex task set. The complex task set consisted of the following:

1. Copy the *package* file in the *business* directory to the *box* file in the *business* directory.
2. Rename the *papers* file in the *supplies* directory to *document* in the *supplies* directory. The *supplies* directory is a subdirectory of the *business* directory.
3. Create a file called *car* in the *ground* directory and sort *ground* files by file size. *Ground* is a subdirectory of *transpor*.
4. Delete the *planes* directory. The *planes* directory is a subdirectory of the *military* directory. This requires deleting all the files in the *planes* directory and then deleting the directory.
5. Find the largest file of **all** the directories and rename the file to *large.fil*. This task requires sorting all the files in all the directories by size to find the *dollar-3* file in the *business* directory and renaming it *large.fil*.
6. Move the *west* subdirectory under the *flags* directory to the *transpor* directory. This requires creating a new directory called *west* under the *transpor* dir and moving **all** the files from the *west* directory under the *flags* directory into it. This could be done one of three ways in the complex interface, but in the simple interface each file had to be

copied and then the empty directory had to be deleted. If the subject using the complex interface made the decision to do so, a simple move command would perform the same function. However it was possible to approach the task in the same manner as in the simple interface.

C. SETTING AND PROCEDURE

Both of the interfaces were selected to work with the Microsoft Disk Operating System (MS-DOS), and both required the same hardware. The simple interface was developed for the Reinhard study (Reinhard, 1991), and the complex interface was Microsoft Windows 3.0 configured in a special set-up. The entire experiment consisted of reading introductory information material, an instruction set for the particular interface in use, then accomplishing two task sets, one simple and the other complex. There was a short questionnaire that the subjects were asked to complete before beginning the experiment, and a longer one they were asked to complete when they had completed the final task set.

The entire experiment was conducted under controlled conditions, in a quiet computer laboratory with no distractions. The machines used were Unisys 386 personal computers configured with math co-processors and VGA color monitors. The individual task sets were conducted on directory systems that were introduced on floppy disks. The subjects had no requirement to access any other disk drive. The maximum number of subjects tested at one time was seven and the least number tested was one. The type of interface was mixed in any group (i.e., there was not interface specific group testing conducted).

Subjects were seated at their machines and encouraged to get comfortable. The test documentation was already placed in front of them. Appendix B is the documentation that was presented to those subjects using the simple interface, and Appendix C was the documentation that was presented to those using the complex interface. The actual test documentation was titled Direct Manipulation Interface Number One and Direct Manipulation Interface Number Two in order to prevent introducing a bias by having the

words "simple" and "complex" appear in the title of the instruction set. The task sets were labeled Practice Task Set, Task Set Number One and Task Set Number Two for the same purpose. It was explained that questions would be answered to clarify a point or offer interpretation, if by doing so it would not detract from the experiment, but instructions pertaining to task accomplishment or interface operation were to be found in the instructions in the documentation provided or in the online help screens.

The first action the subjects took was to read a short introduction to the experiment and a privacy act statement. Then they were asked to complete the Verbalizer-Visualizer Questionnaire to characterize their thinking style as visual or verbal (Richardson, 1977). They next were presented with basic information on file management, directory structures, the operating system they were to be using and the tasks themselves. The instruction sets for the two interfaces were the same. The only differences were the instructions for the interfaces and the procedures for data collection. Information provided on interface operation was specific to the interface the subject would be using.

Once the subjects had started the testing, they were asked to continue through to the end without stopping. All of the subjects were able to accomplish the entire experiment in one sitting.

After reading the preliminary instructions, the subjects then were asked to work on the practice set. The practice set included operations to access the online help feature, create a file, copy a file, sort files, delete a file and rename a file. For directory operations the practice set required the subject to create a directory, delete a directory, and show the directory tree. The user was asked to work through these steps until comfortable with the interface. The subject was also asked to experiment with the interface to get a good feel for the functionality, not being limited to the specific steps in the practice set. Subjects were encouraged to repeat the practice sets if they did not feel comfortable with the interface after one run through. The average time spent training and practicing the simple interface was 20.3 minutes with a range of 42.7 to 7.5 minutes. The average time spent training and practicing the complex interface was 24.7 minutes with a range of 38.25 to 13.3 minutes. In the Reinhard experiment, the average time

spent training and practicing for the direct manipulation interface was 42.5 minutes with a range of 25.6 to 74.4 minutes. One significant difference in the tested population was that Reinhard's subjects were all novice computer users.

During the practice set subjects proceeded at their own pace and were encouraged to attempt to do all the tasks without help from the experimenter. Although a visible effort was made to keep interventions to a minimum, subjects were instructed to call the experimenter for assistance if they had tried but were were making no progress in the resolution of a problem.

After the subjects had worked all they wanted on the practice set, they had to call the experimenter to set up the interfaces for the simple task set. This procedure called for the experimenter to install the directory system for the next task set, which was completely different than the one used for the practice set.

The subjects had to call the experimenter between the completion of the simple task set and the start of the complex task set so that the experimenter could set up the interface. Once again this procedure called for a complete change of the directory system.

When the subject had finished all three task sets, a questionnaire was completed. The questionnaire contained questions asking about the interface they had just used, questions about the amount of experience with operating systems they have had, and questions to determine the level of computer sophistication they possessed. The questionnaire is found in the last pages of the instruction sets in Appendices B and C.

D. APPARATUS

1. Simple Direct Manipulation Interface

The same direct manipulation interface developed for Reinhard's study was used in this study as the simple interface for all three task sets. The interface was developed using **Smalltalk/v**, an object oriented programing system[2]. Subjects using this

[2] Smalltalk is a product of Digitalk, Inc.

interface were given the Direct Manipulation Interface Number One instruction set (Appendix B). This instruction set contained a description of operating systems, file management, and an operational description of the interface the subjects were to use.

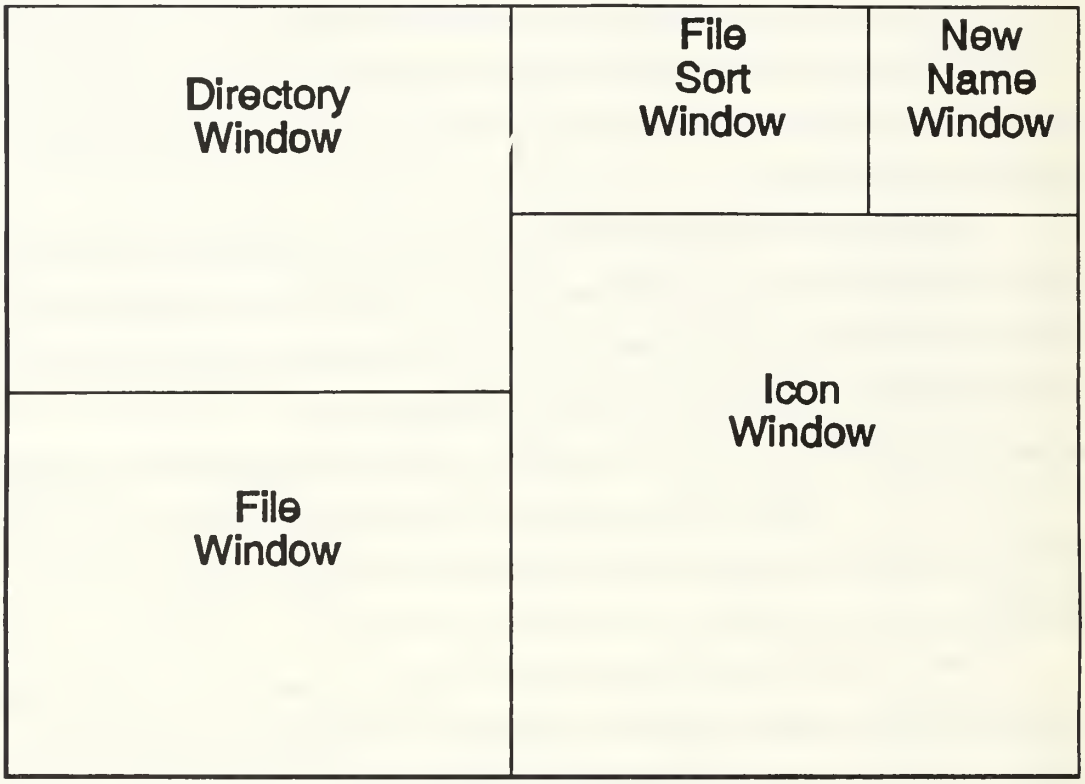

Figure 1
Simple Interface Window Structure

Figure 1 shows the basic window structure for the interface. The structure contains five windows: 1) the Directory Window, which displays the hierarchical directory structure, 2) the File Window, which displays the files listed in a specified directory, 3) the File Sort Window, which allows the user to sort files by name, date of file creation, or file size, 4) the New Name Window, which contains all allowed names needed for new files to be created, renamed files, and new directories, and 5) the Icon Window, which contains all the icons used for tasks operations. Figure 2 is a representation of an actual interface screen.

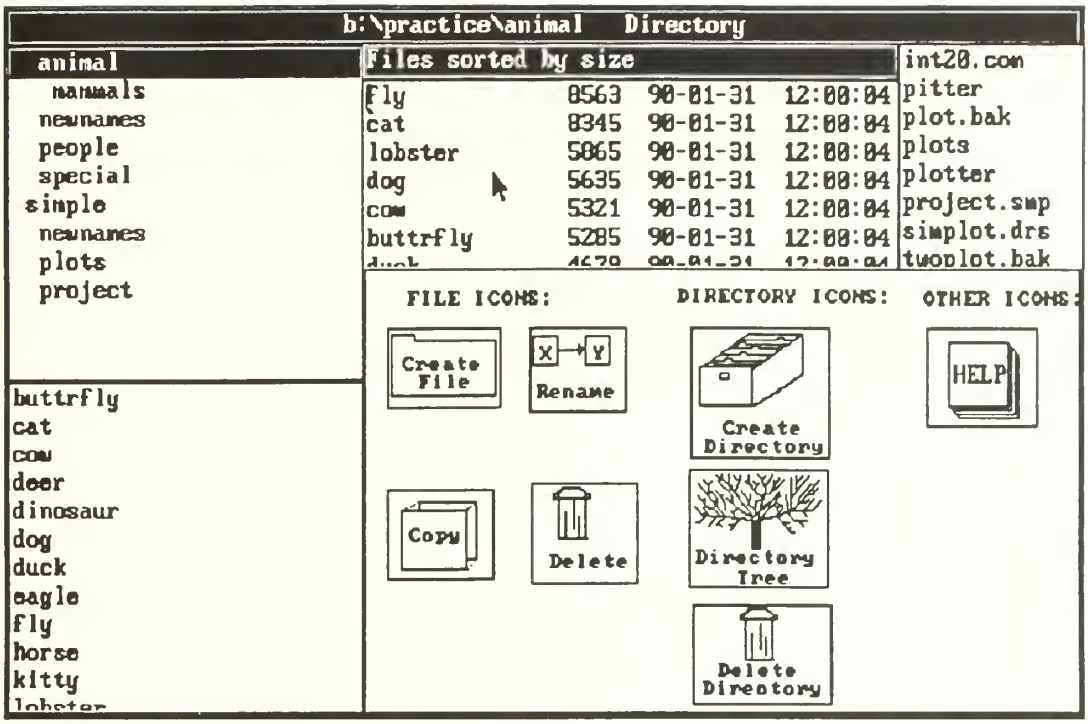

Figure 2
Simple Interface Opening Screen

Figure 3 shows what the Help Window looks like on the screen when the Help Icon is selected. The user then selects the item of interest from the listing of topics on the left side of the window by placing the mouse pointer on it and clicking the left mouse button. The right hand side of the window will then display the step-by-step procedural information requested

by the user. Figure 3 shows the Help Window with the Create Directory file topic selected. Figure 4 shows the directory tree that is displayed when the Directory Tree Icon is selected. The Directory Tree Window then displays the names of all the subdirectories contained on the drive selected, in this case the floppy disk which contained the directory system used for the experiment.

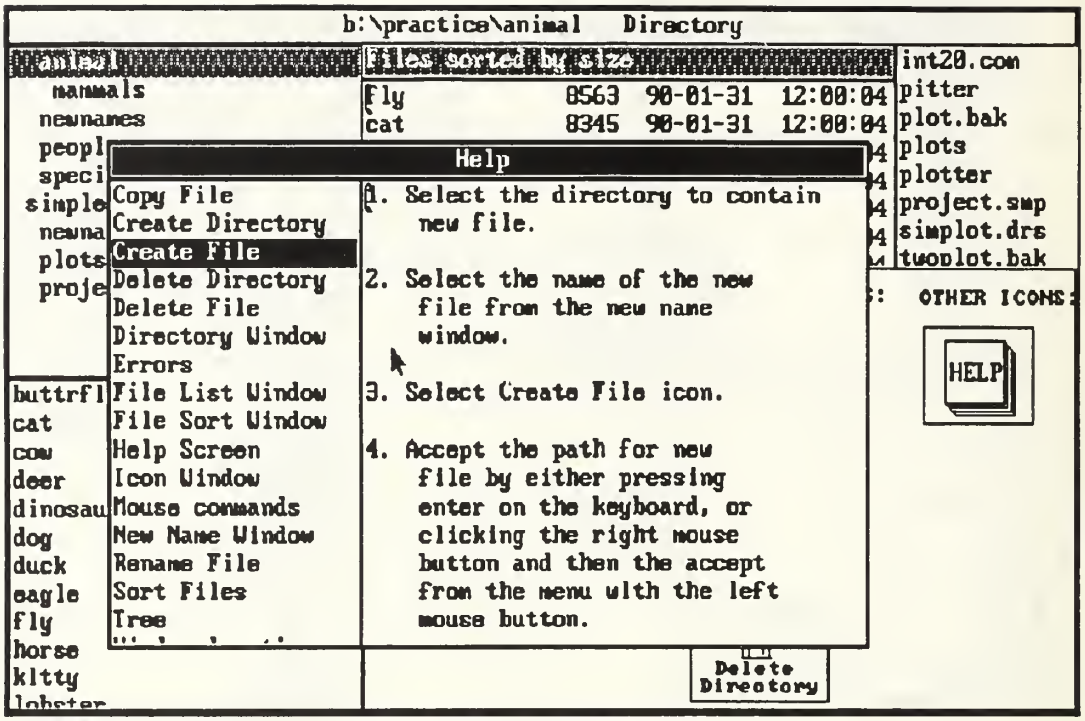

Figure 3
Simple Interface Help Window

When the subject properly completes the operation to accomplish a task on a file or a directory, a Prompter Window appears allowing the user to accept or cancel the current operation. An example of this Prompter Window is shown in Figure 5.

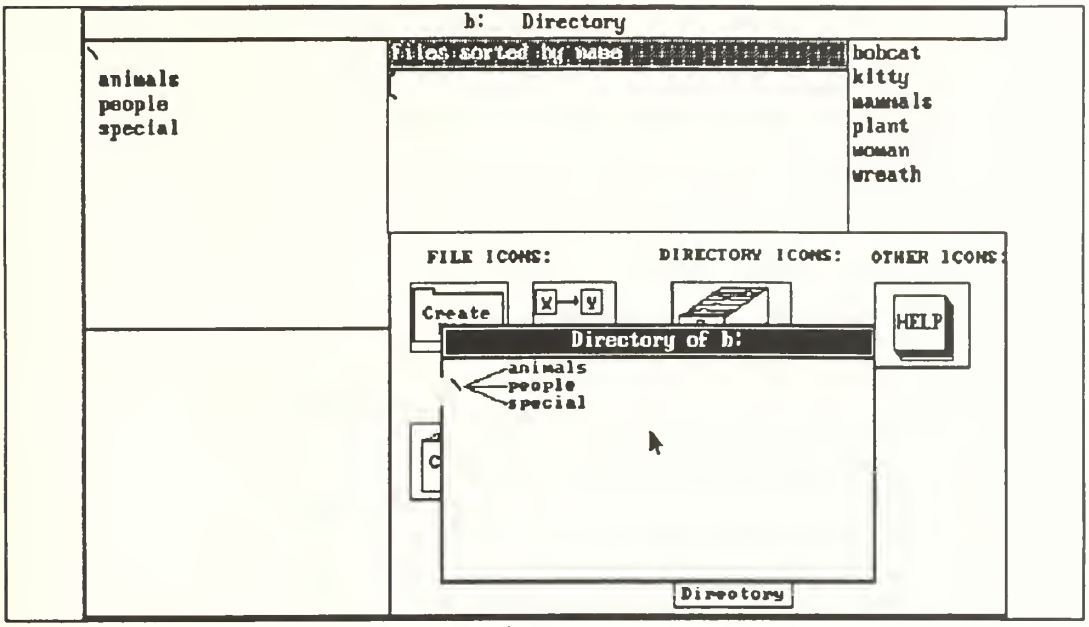

Figure 4
Simple Interface Directory Tree

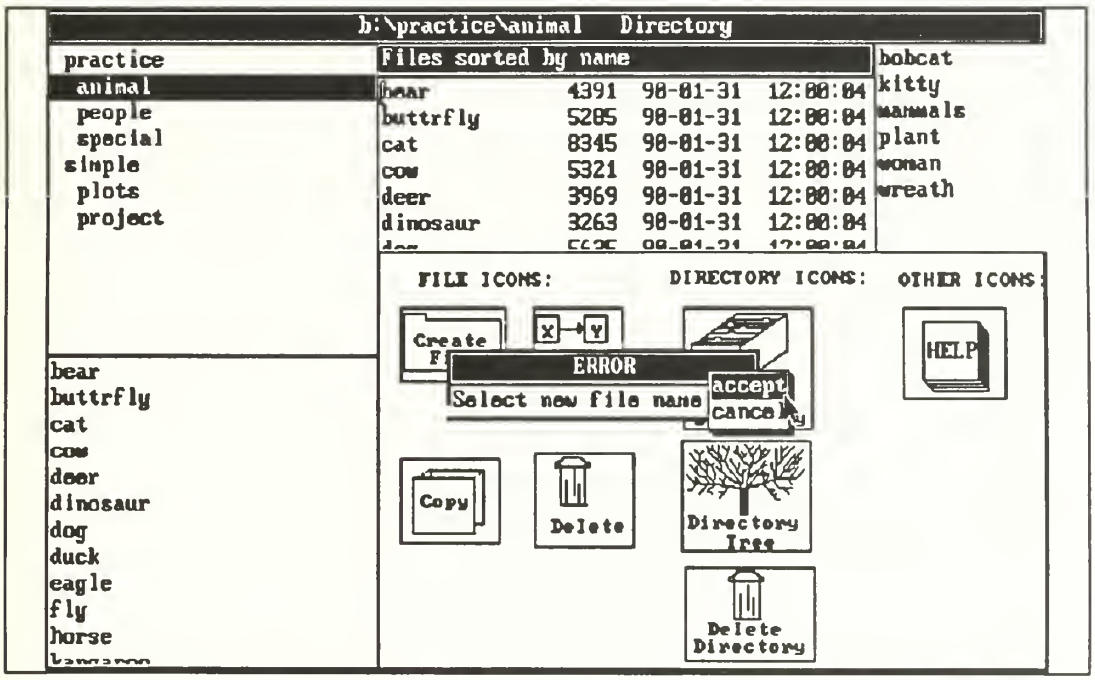

Figure 6
Simple Interface Error Message

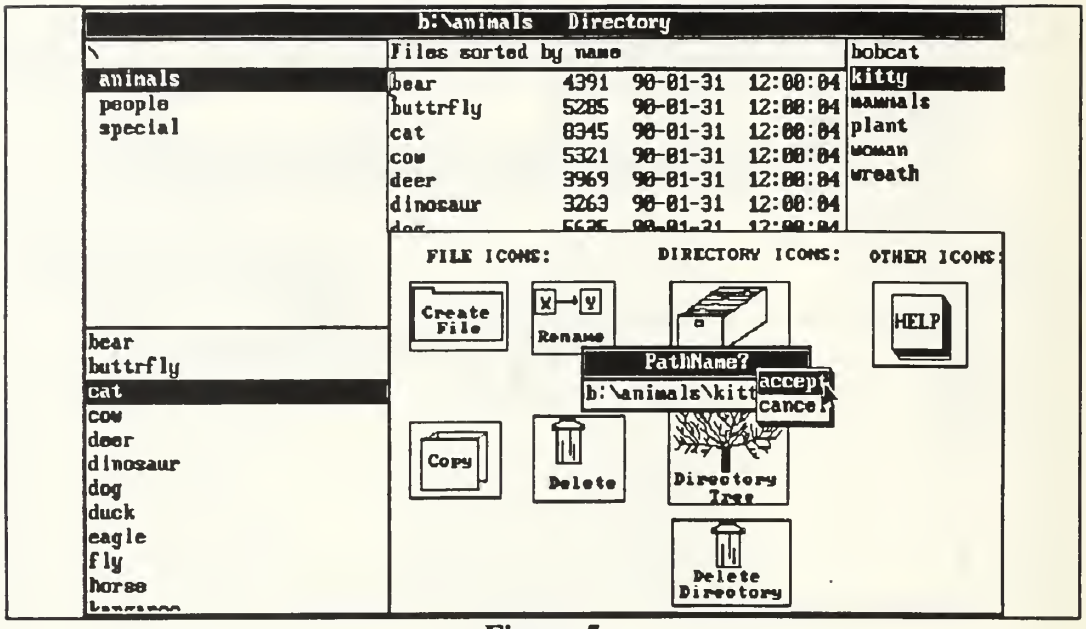

Figure 5
Simple Interface Prompter Window

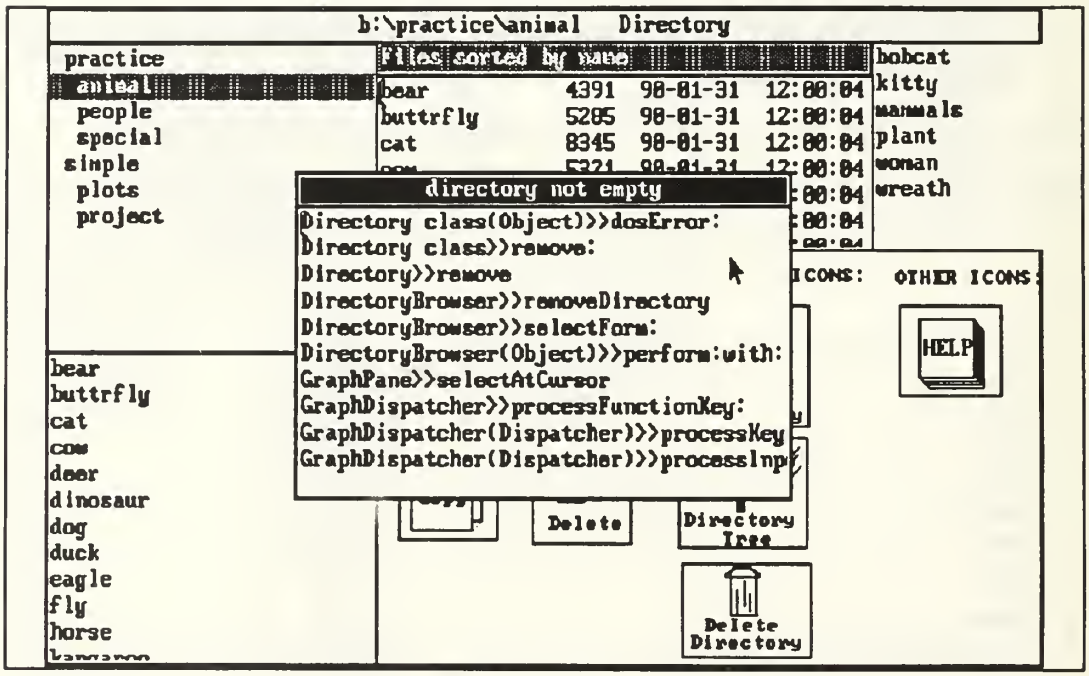

Figure 7
Smalltalk Error Window

If the operation was not conducted properly and insufficient information is provided, a Prompter Window is displayed with an error message to the user. The error message will identify what has been done incorrectly, but information necessary to correct the condition is not provided by the error message. An illustration of an error message is shown in Figure 6. If the subject attempts to perform an operation that is not allowed by the operating system, an Error Window with a short message in the label is displayed in the center of the display screen. The error message is generated by **Smalltalk**. The **Smalltalk** code is displayed in the lower portion of the window, as seen in Figure 7. This window is used to ascertain the reason for the error shown in the label. The example shown was the result of an attempt to delete a directory that contained either a subdirectory or contained one or more files.

a. Practice Task Set Screen

Figure 8 shows what the screen looks like for the practice task set. The subject has selected the *animals* directory for display, which lists all of that directory's files. The subject has also selected the name *kitty* from the New Names Window, and the *cat* file from the File Window.

b. Simple Task Set Screen

Figure 9 is an example of the screen for the simple task set. The user has selected the *project* directory. The files for this directory are shown in the File Sort Window and the File Window.

2. Complex Task Set Screen

Figure 10 is an example of the complex task set screen. The subject has selected the *business* directory, and the files in that directory are displayed.

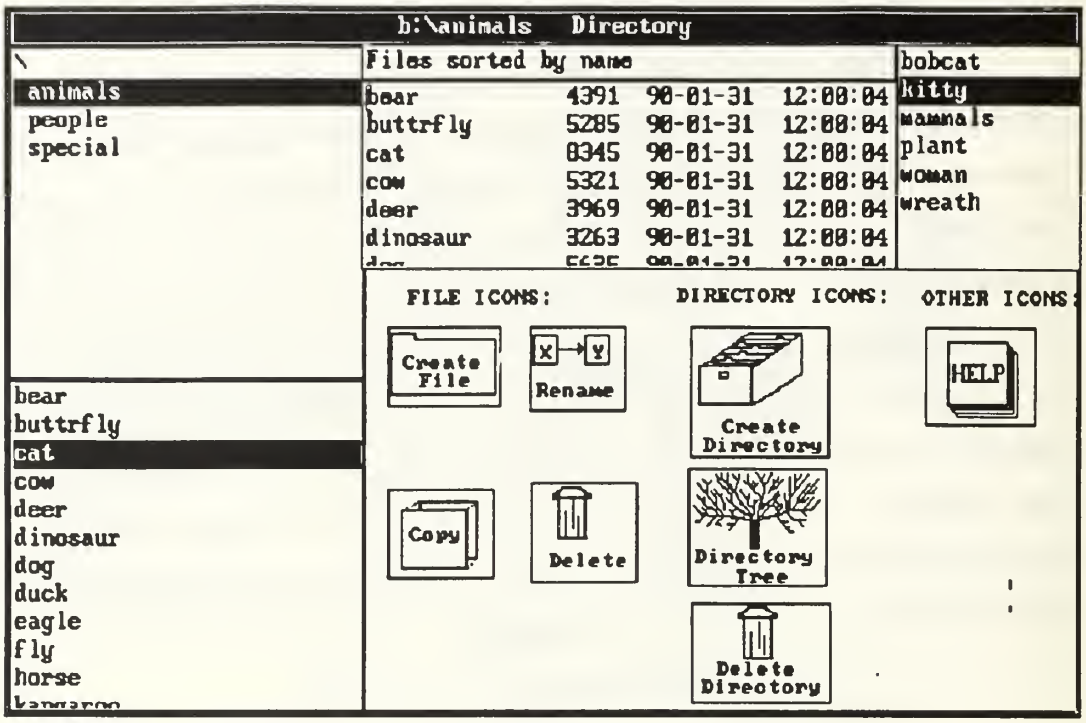

Figure 8
Simple Interface Practice Task Set Screen

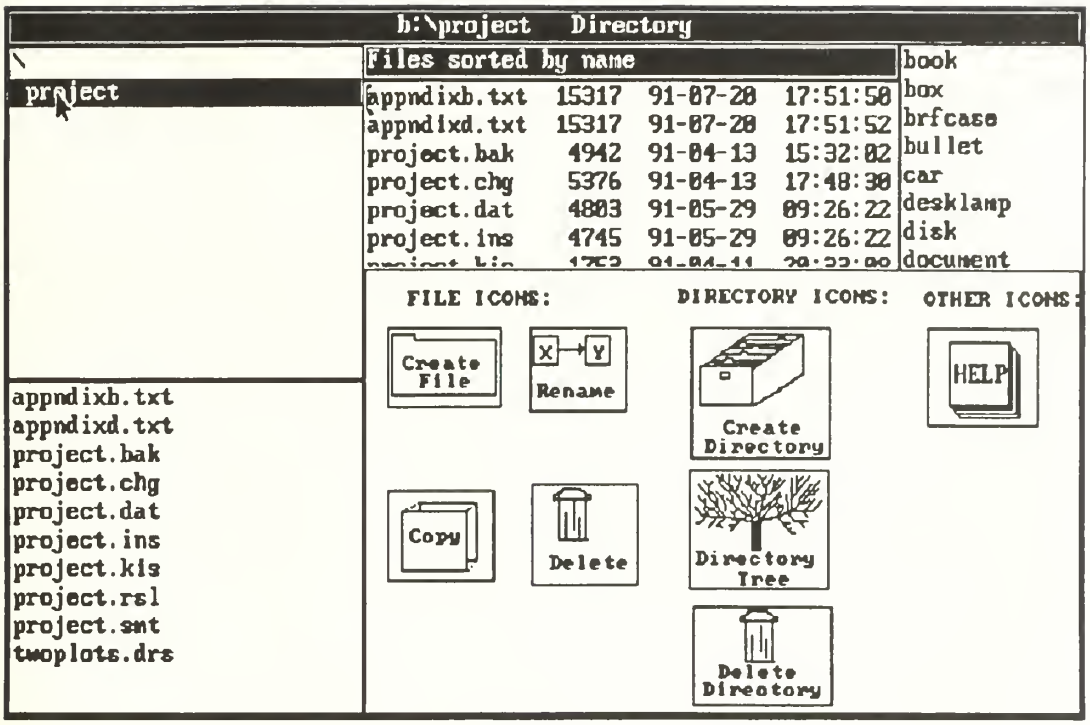

Figure 9
Simple Interface Simple Task Set Screen

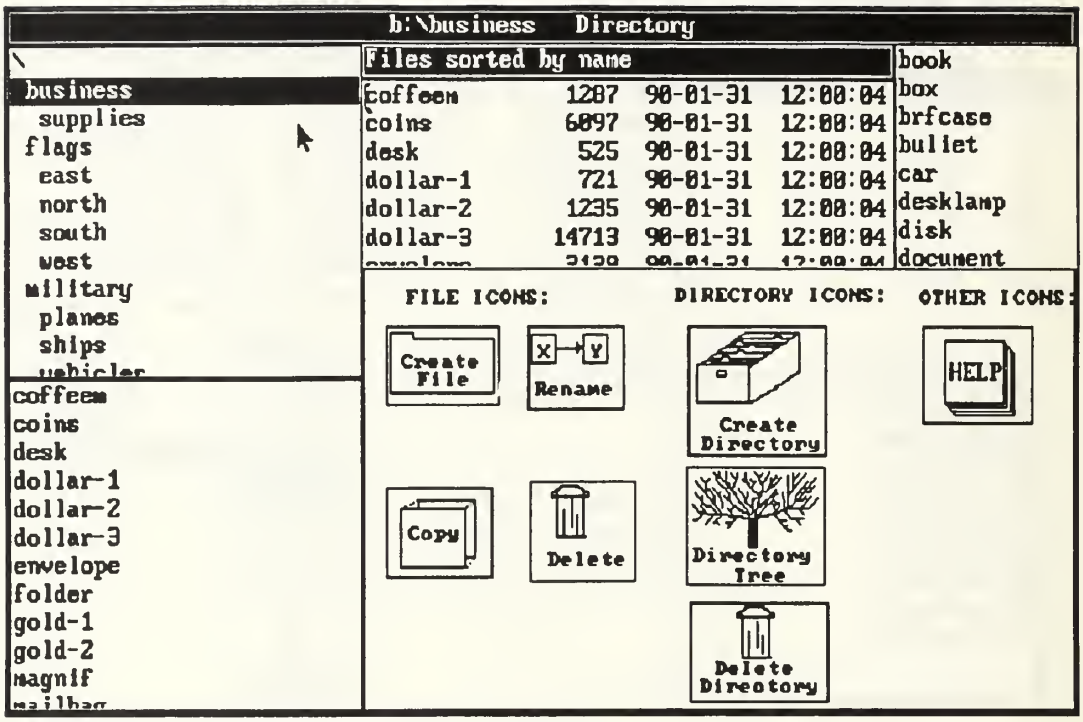

Figure 10
Simple Interface Complex Task Set Screen

3. Complex Direct Manipulation Interface

The complex interface was derived from a commercially available product called Windows 3.0[3]. This interface was chosen because it is a direct manipulation interface and it has a very wide range of functions. Subjects using this interface had no prior experience on the Windows interface. Subjects using this interface were given the Direct Manipulation Interface Number Two instruction set (Appendix C). This instruction set included the same introduction material as for the simple interface instruction set. It also included the operating instructions necessary for the subjects to accomplish the tasks assigned. Windows was developed to be a fully functional DOS shell, but the subjects were restricted in using only the file management functions, which fully supported all three task sets. Most selections were made as in the simple interface, **pointing** and

[3] Windows 3.0 is a registered trademark of the Microsoft Corporation.

clicking the left mouse button. Opening a directory, however, required the subject to double click the left mouse button. Figure 11 shows what the opening screen for the complex interface looked like.

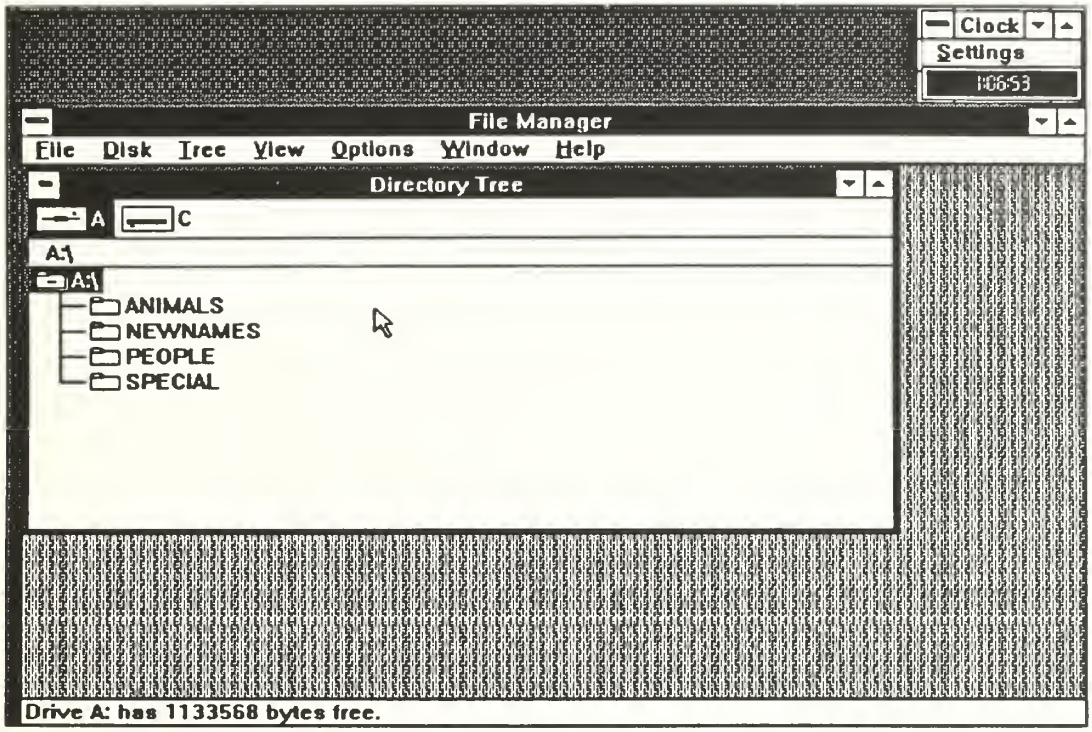

Figure 11
Opening Screen for the Complex Interface

Figure 12 shows the menu that is displayed when help is selected. Help is selected by putting the pointer on Help and clicking the left mouse button. The desired menu item is then selected and the action is completed. Figure 13 shows the result of selecting the Index option from the Help Menu. The displayed listings have been scrolled through about half way, as shown by the scroll bar on the right hand window frame. All windows that can be scrolled are done so by selecting either the up or down arrows in the small boxes above and below the scroll bar with the mouse and holding down the left button until the item of interest is displayed in the window.

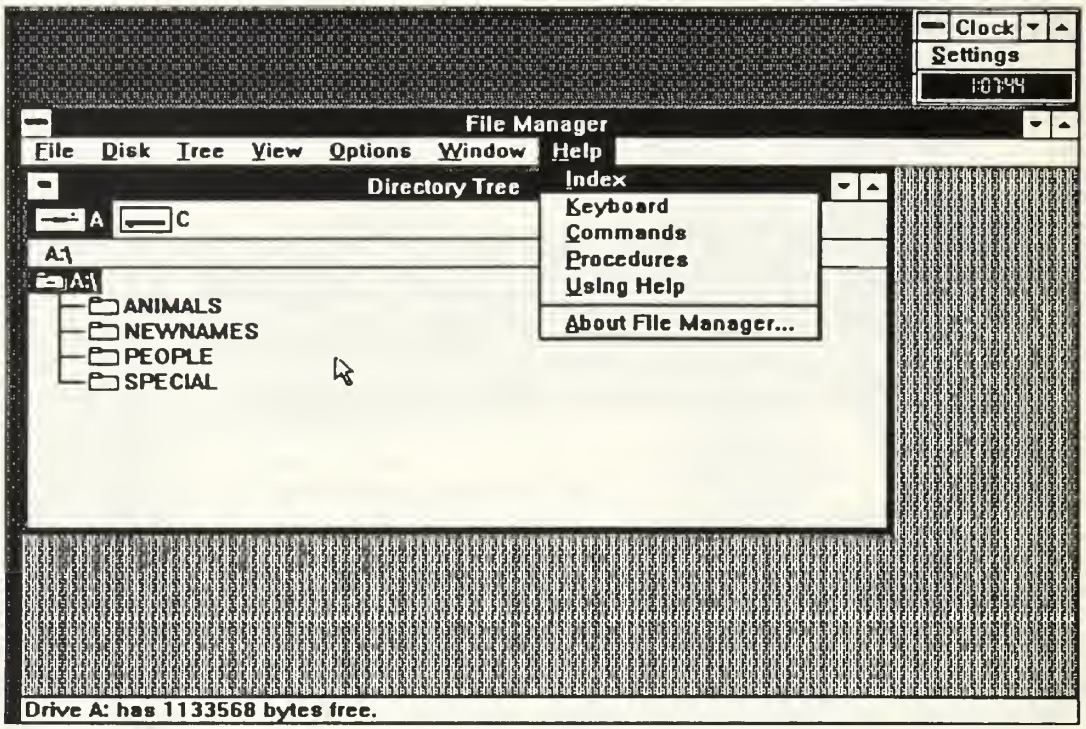

Figure 12
Complex Interface Help Menu

When the user performs an operation on a file or directory, one possible outcome is the path window shown in Figure 14. Whenever the path is required as part of the operation to accomplish the task, this window will appear. The user has the opportunity to define the path, as is the case in Figure 14, or change the path statement.

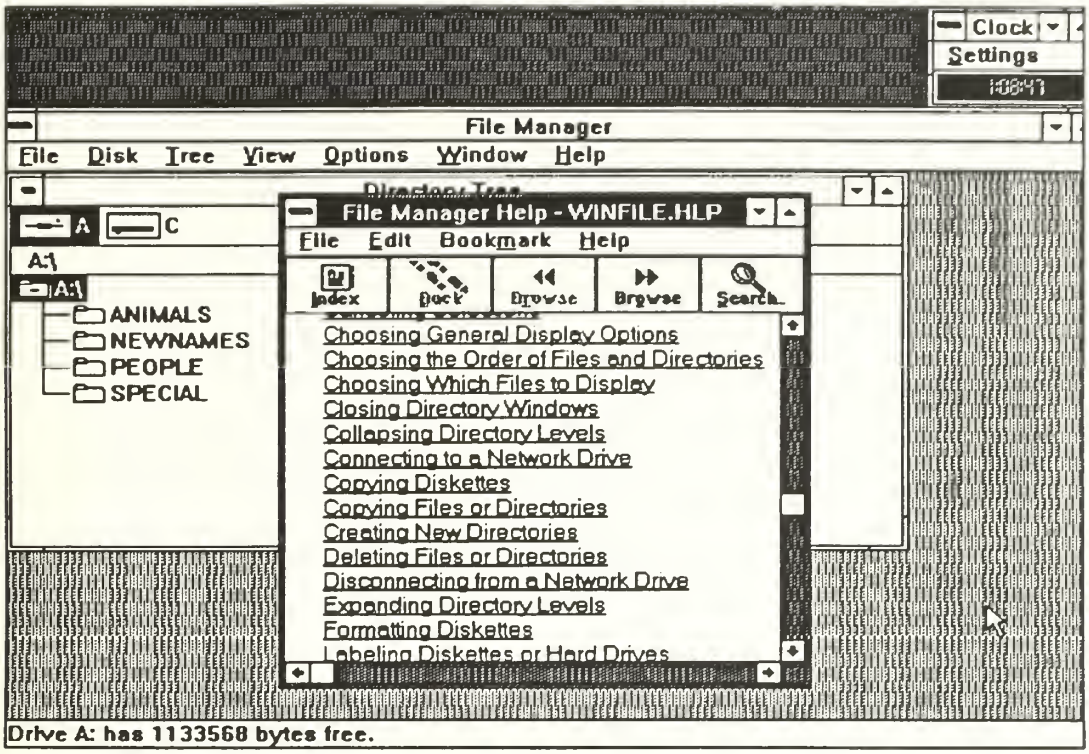

Figure 13
Complex Interface Help Window

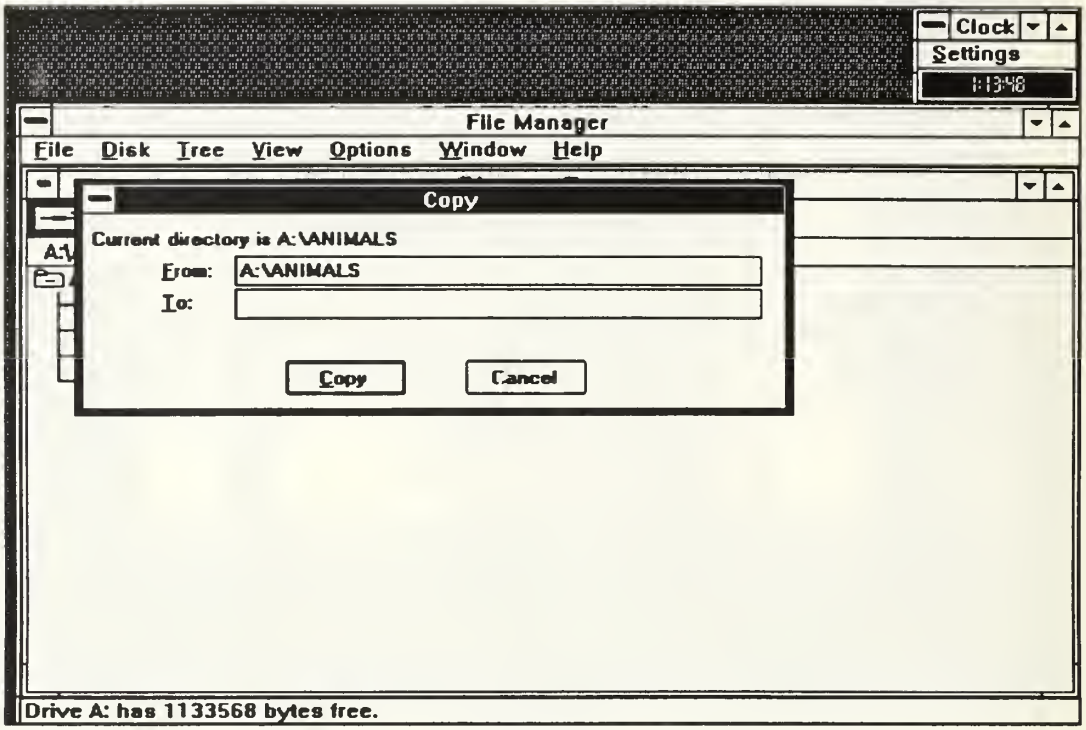

Figure 14
Complex Interface Path Window

If the subject performs an improper operation, an error window appears, as in Figure 15. Information in the Error Window describes the error or informs the user why the operation cannot be accomplished as attempted. Errors that occur because of an incorrect operation of the interface as well as errors that are a result of a violation of some rule of the operating system are treated in the same manner.

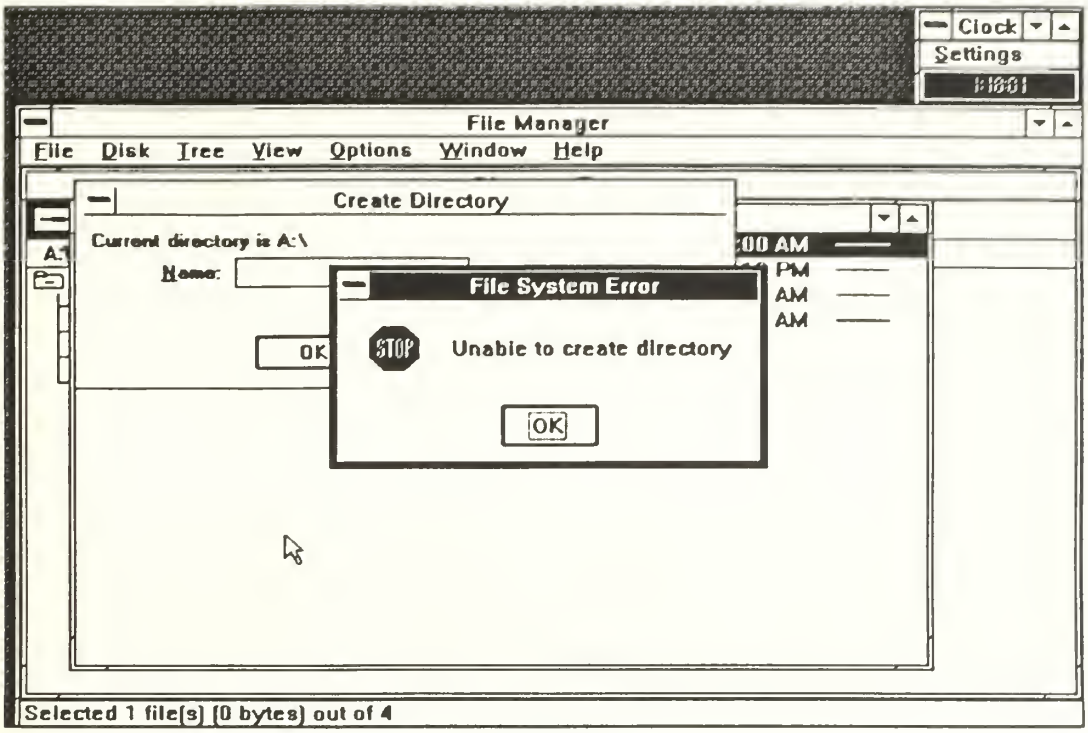

Figure 15
Complex Interface Error Window

Figure 16 shows the menu that is displayed when the "minus" sign in the small box in the upper left hand corner of the windows is selected. This menu is called the Control Menu, and it allows the window to be closed as one of its options.

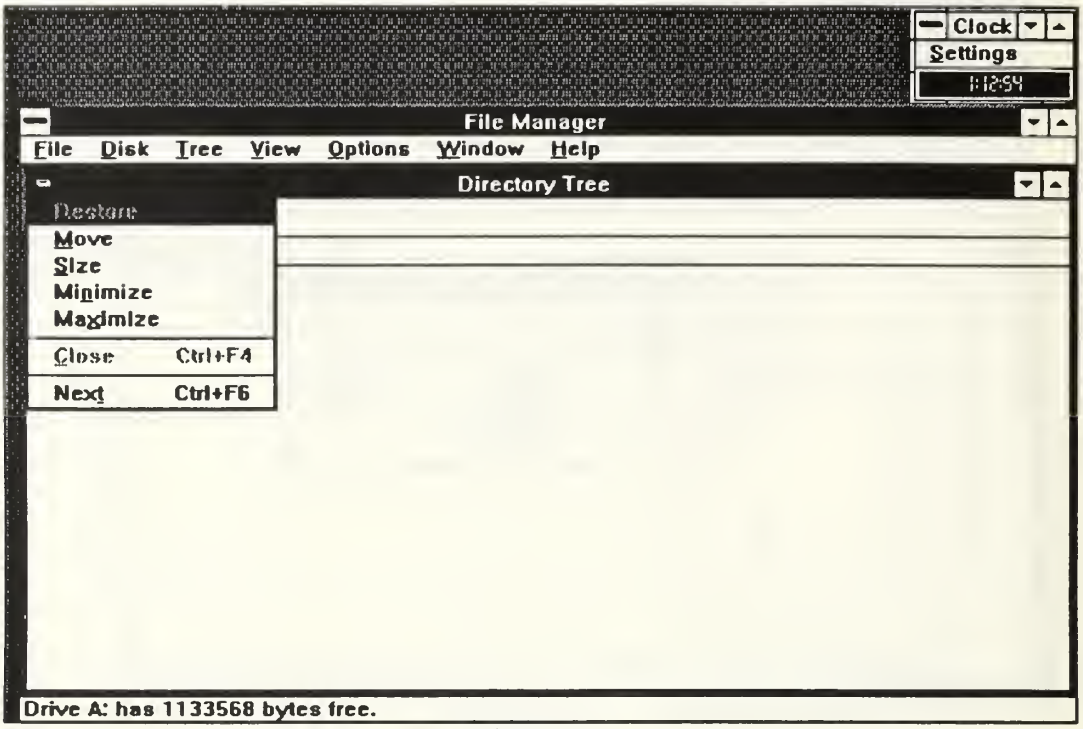

Figure 16
Complex Interface Control Menu

Figure 17 is an illustration of the View Menu. This is the menu to select for the file sort operation. The Sort Window is shown in Figure 18.

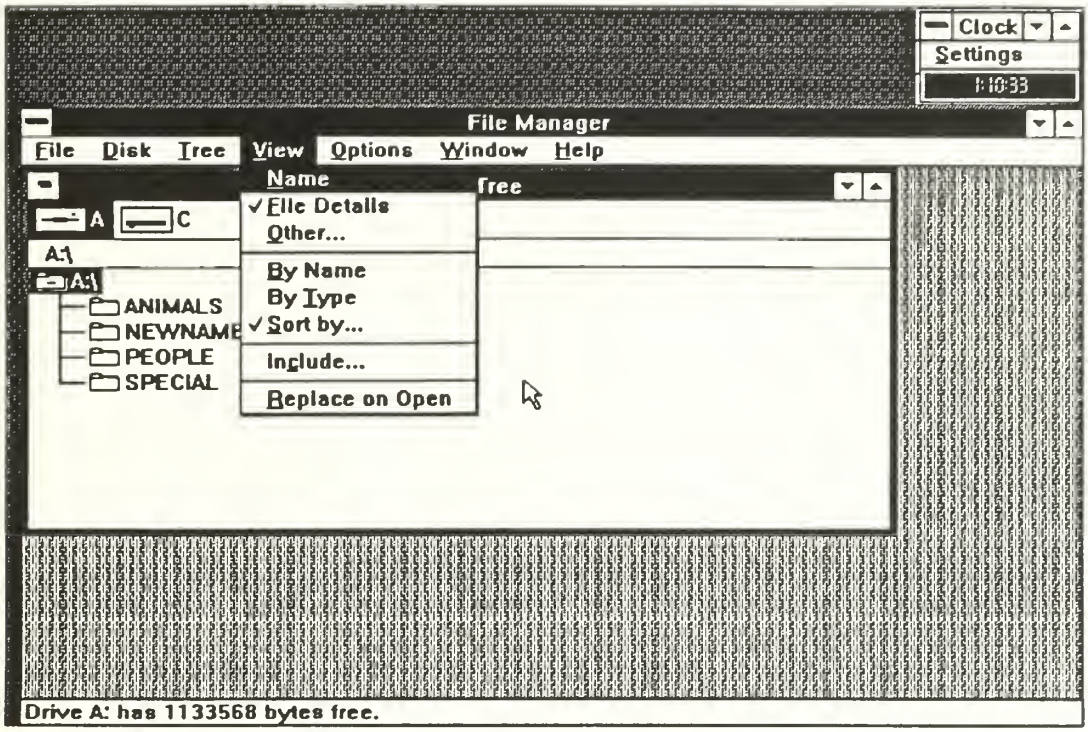

Figure 17
Complex Interface View Menu

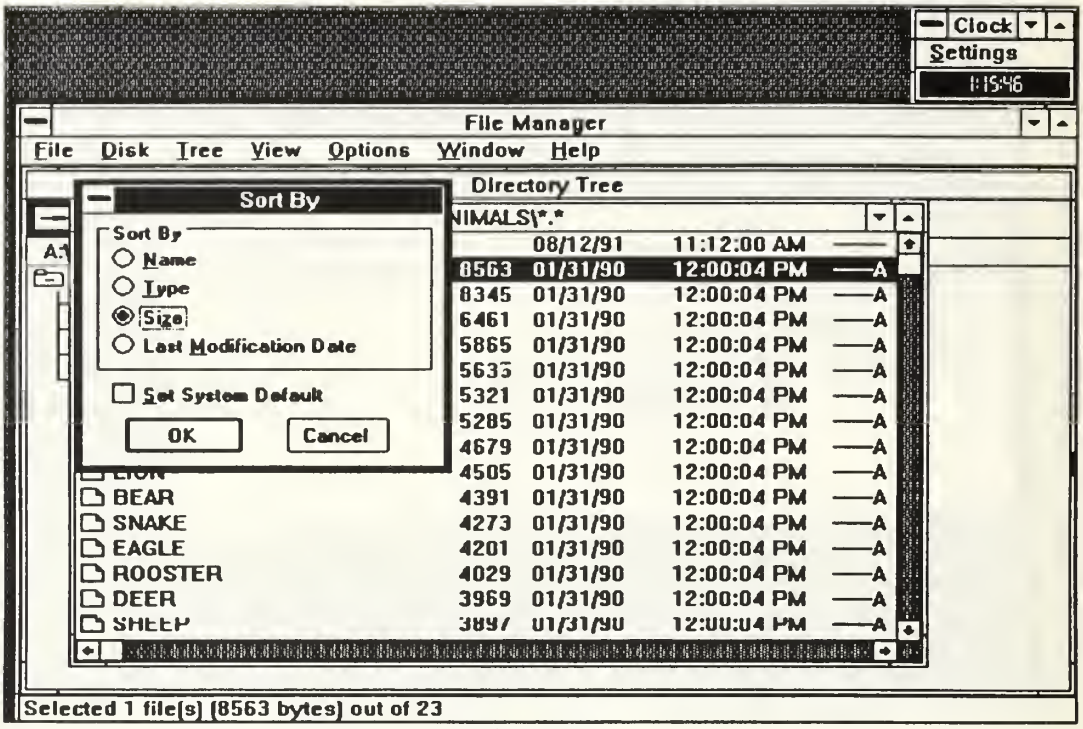

Figure 18
Complex Interface Sort Window

Figure 19 shows the File Menu. This is the menu which contains the operations necessary to do the create, copy, delete, and move operations required by the task sets.

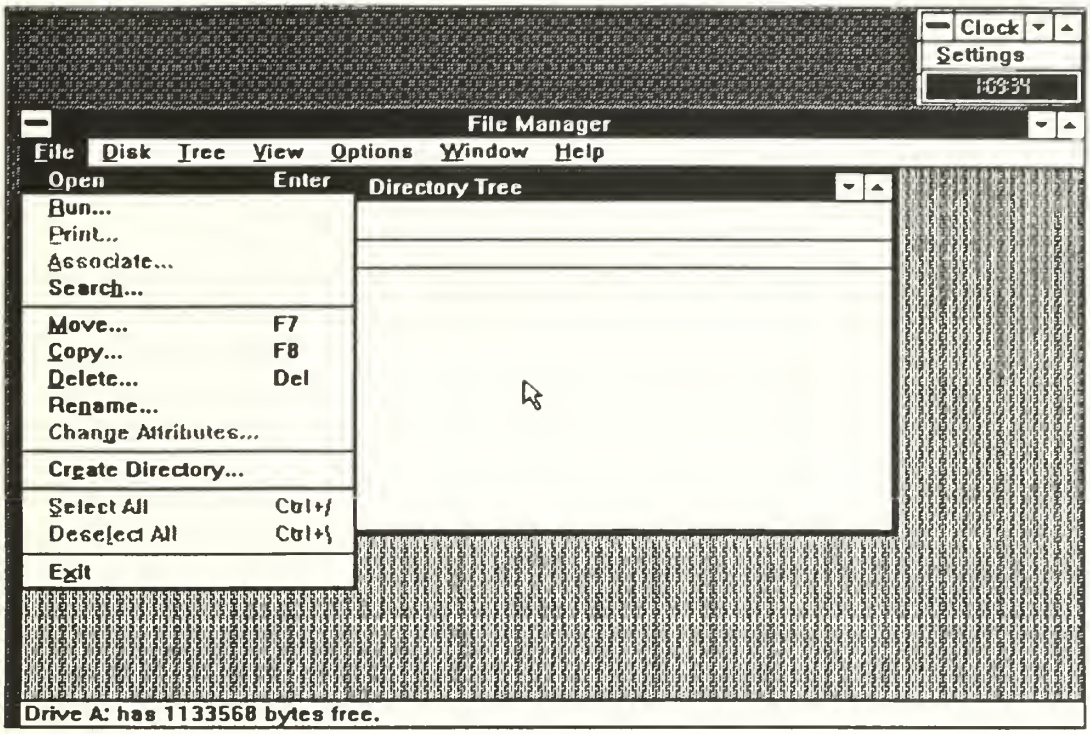

Figure 19
Complex Interface File Menu

Figure 20 shows the Window Menu. This menu has as an option the feature to show concurrently all directories that are opened. This was a great help in the directory search for the largest file.

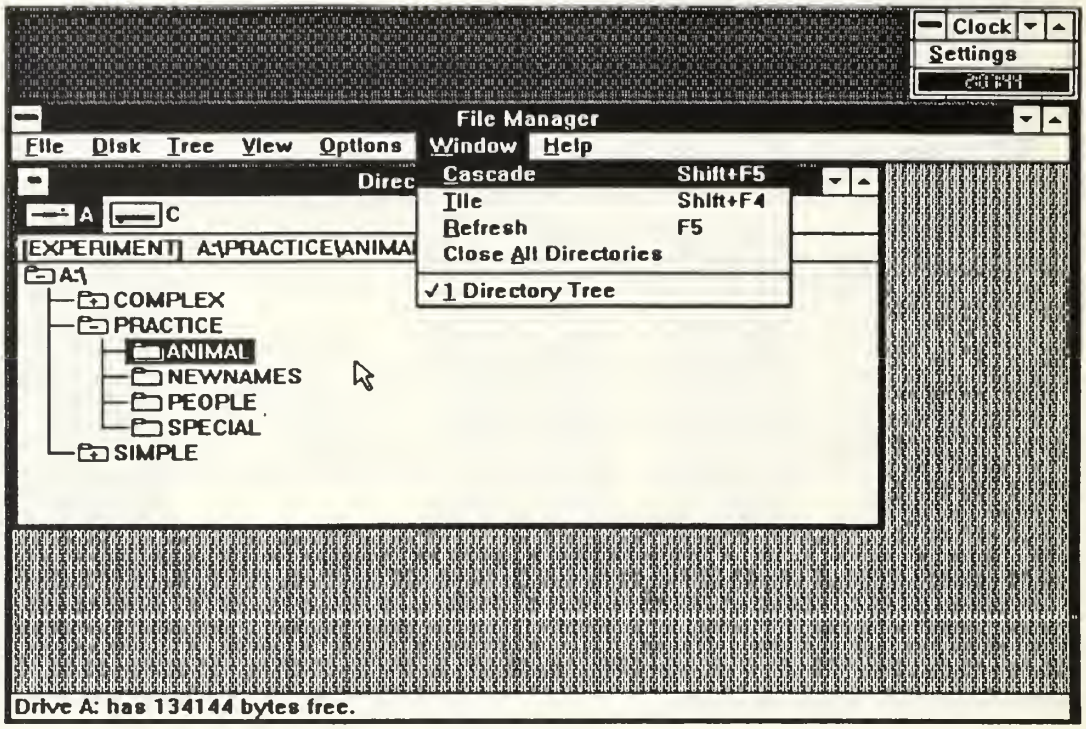

Figure 20
Complex Interface Window Menu

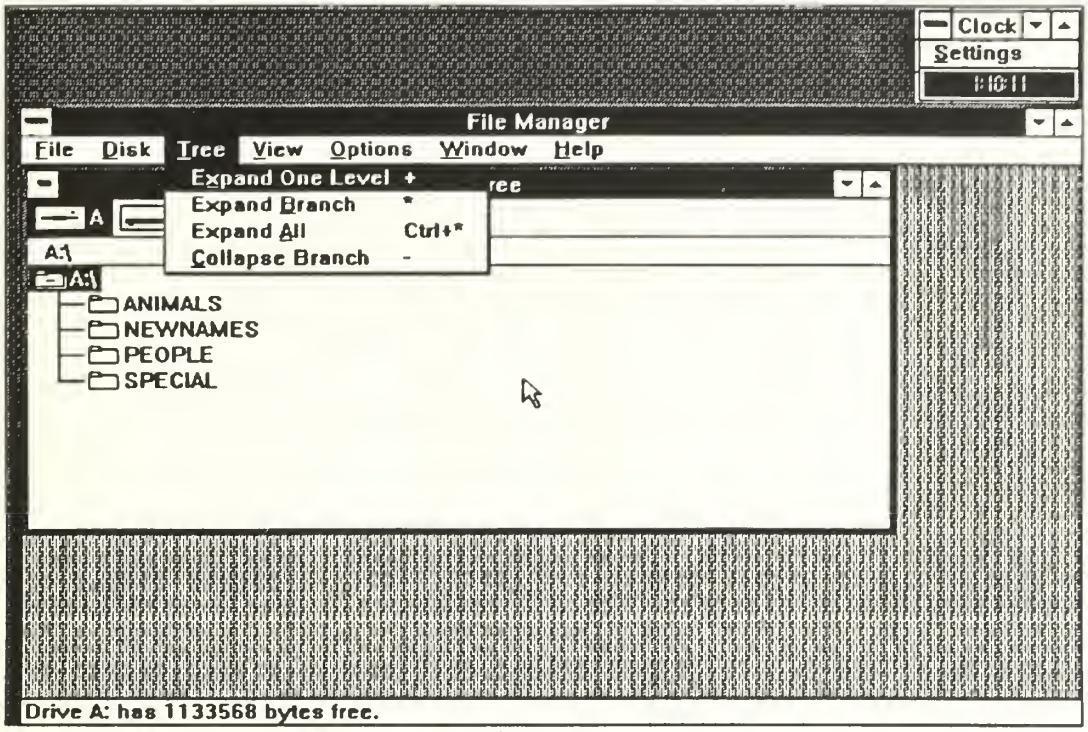

Figure 21
Complex Interface Tree Menu

Figure 21 shows the Tree Menu. This gives the user the option of expanding a selected branch of the directory tree or the entire tree. It is a helpful option to be able to maintain the basic tree structure on the screen while allowing the user to expand a single branch of interest.

The complex interface also provided prompter windows to allow the subject to approve an operation before the interface accomplished it. Figure 22 shows the prompter window that is displayed when a file deletion operation is being attempted.

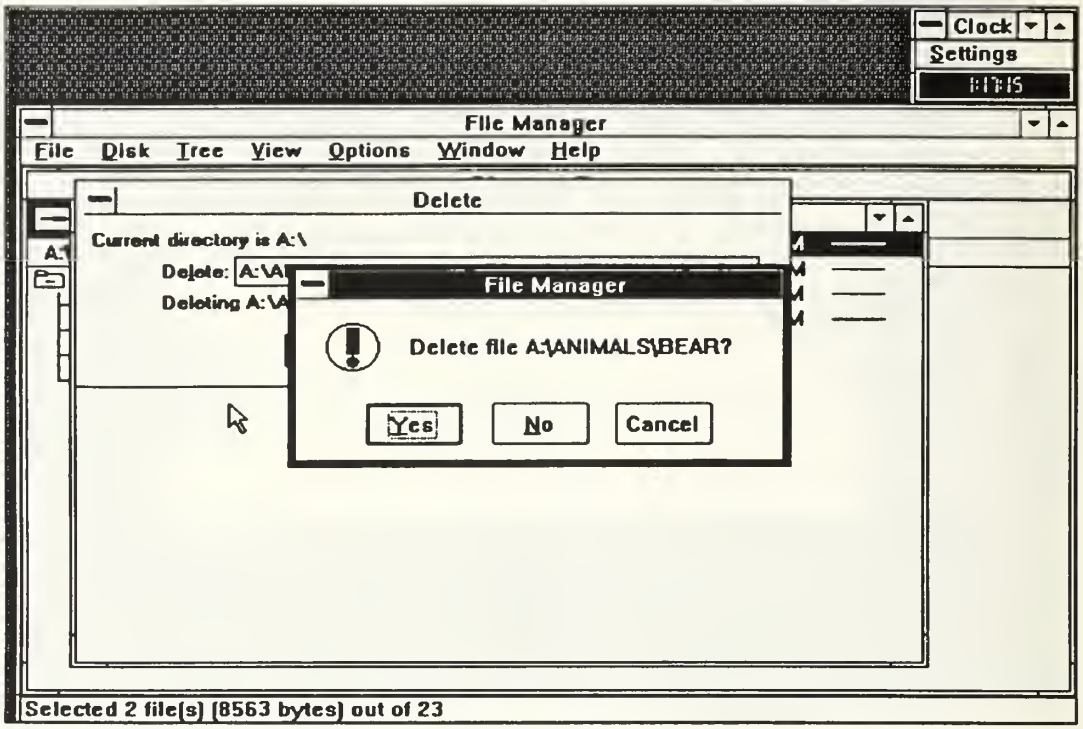

Figure 22

Complex Interface Prompter Window for a File Deletion Operation

a. Practice Task Set Screens

Figure 23 shows what the complex interface screen looked like in the practice task set. The ANIMALS directory has been selected and the directory has been opened, showing the files contained in that directory in the directory window.

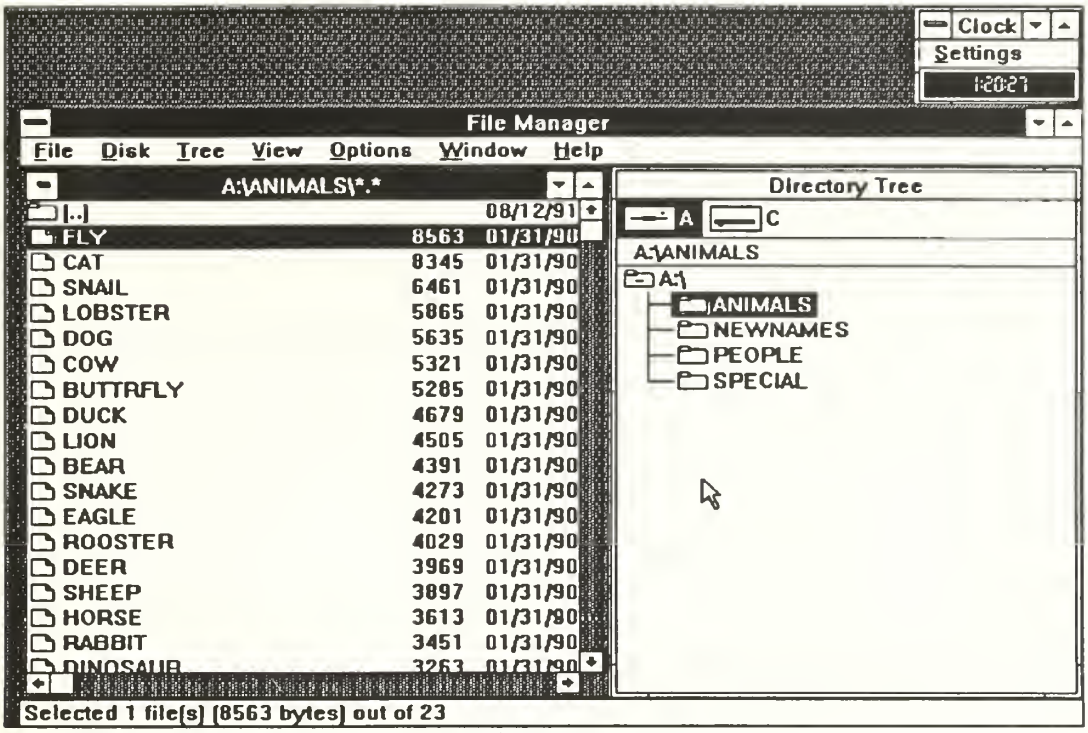

Figure 23
Complex Interface Practice Task Set Screen

b. Simple Task Set Screens

Figure 24 is showing the complex interface's simple task set screen, with the PROJECT directory opened and the files in the directory displayed in the directory window.

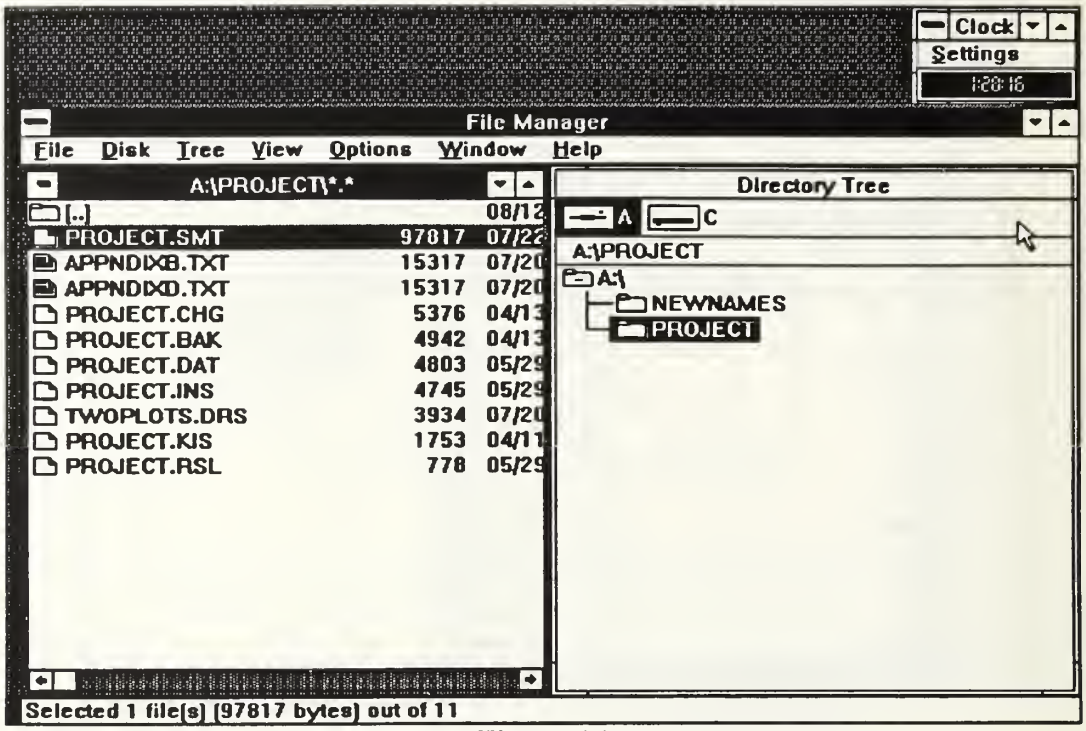

Figure 24
Complex Interface Simple Task Set Screen

c. Complex Task Set Screen

Figure 25 shows the complex interface complex task set with the BUSINESS directory selected and the files and subdirectories contained in that directory.

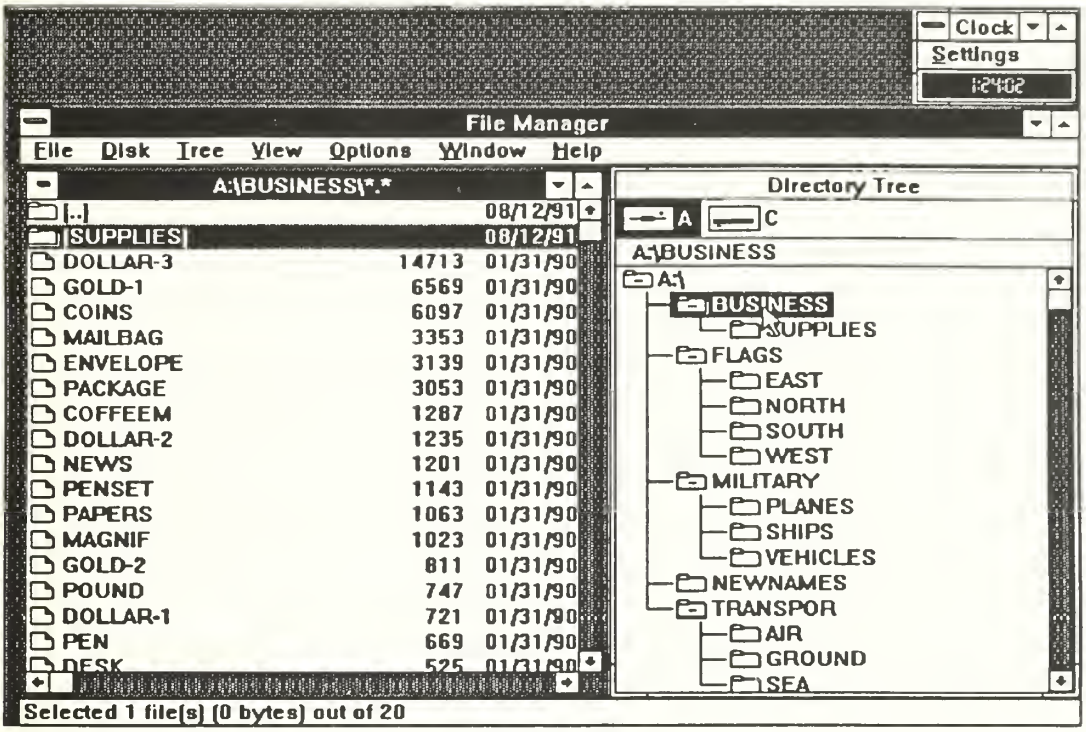

Figure 25
Complex Interface Complex Task Set Screen

E. DEPENDENT MEASURES

1. Primary Measures

The dependent variables were also the same measurements of performance used in the original experiment. These variables were (1) time required to complete each task set, (2) the number of errors committed during the execution of each task set, and (3) the number of times the on-line help function was called.

2. Secondary Measures

As in the original, an attempt at determining the subject's thinking style was also made. The same Verbalizer-Visualizer Questionnaire (VVQ) test was used to determine the degree the subject stored and accessed pictorial or verbal representations during the accomplishment of the task. There are fifteen questions on the VVQ, each pertaining to pictorial or verbal thought process. The scores were on a scale between 1 and 15, with the low scores indicating a strong verbalizing tendency and the high scores indicating a strong visualizing tendency (Richardson, 1977: pp.109-124).

3. Subject Data Files

In the simple interface a data file was maintained for each subject automatically. The data recorded was the time and actions performed by the subject for each task set. This data file was then analyzed to summarize all times required to complete each task set, the number of errors committed, and the number of times the online help function was accessed. Appendix D is a sample data file from a simple interface task set.

The coding in the data file represents all possible actions that the interface supported. The code explanations are found in Appendix E. If an operation was conducted correctly, the time and the type of operation, including the directory and file name, were recorded. If the operation was not conducted correctly, an error resulted. The time, the operation code, and a description of the error were recorded. All user logs were reviewed manually to ensure all errors were properly identified.

IV. RESULTS

A. DATA ANALYSIS

Data were analyzed using a multivariate analysis of variance (MANOVA) for repeated measures to examine the interrelationships of the dependent variables of completion time, errors, and on line help references (Bray and Maxwell, 1985). The model for the analysis was derived from Winer (1971:pp. 630-633). A univariate analysis of variance was conducted on each dependent variable as well.

There were a total of 29 subjects, 15 using the complex interface and 14 using the simple one. Since the cell sizes were unequal, the General Linear Model Procedure of the SAS statistical software program[4] was used to conduct the analysis.

B. EXPERIMENTAL RESULTS

The means and standard deviations for each of the three dependent variables are given in Table 2. The mean values for these variables are plotted as a function of interfaces and task types in Figures 26 through 28. The graphs indicate that as the task complexity increases, the individuals using the simple interface will require more time for completion and will make more errors. The number of references made to the help function increase from the simple to the complex task set. The subjects using the complex interface required about the same amount of time to complete the complex task set as they did to complete the simple task set, but the number of errors decreased as the complexity of the task set increased. The number of help references remained essentially constant from the simple to the complex task set.

---

[4] SAS is a product of the SAS Institute, Inc. of Cary NC.

Table 2
SUMMARY OF RESULTS
MEANS
(STANDARD DEVIATIONS)

Dependent Variables	Interface Type					
	Simple			Complex		
	Practice Session	Simple Task	Complex Task	Practice Session	Simple Task	Complex Task
Completion Time, Min	20.3178 (10.6473)	10.3857 (6.1941)	25.3857 (7.6023)	24.6912 (6.3637)	20.3400 (9.4520)	20.3922 (7.2081)
Number of Errors	2.2857 (1.9386)	3.3571 (3.5649)	7.1429 (6.3833)	14.8667 (8.4165)	10.0000 (9.5093)	4.4667 (4.5177)
Number of Help References	2.5714 (2.0649)	1.0714 (1.7305)	2.7143 (2.2336)	1.7333 (1.0330)	2.6000 (1.8048)	2.2667 (2.3135)

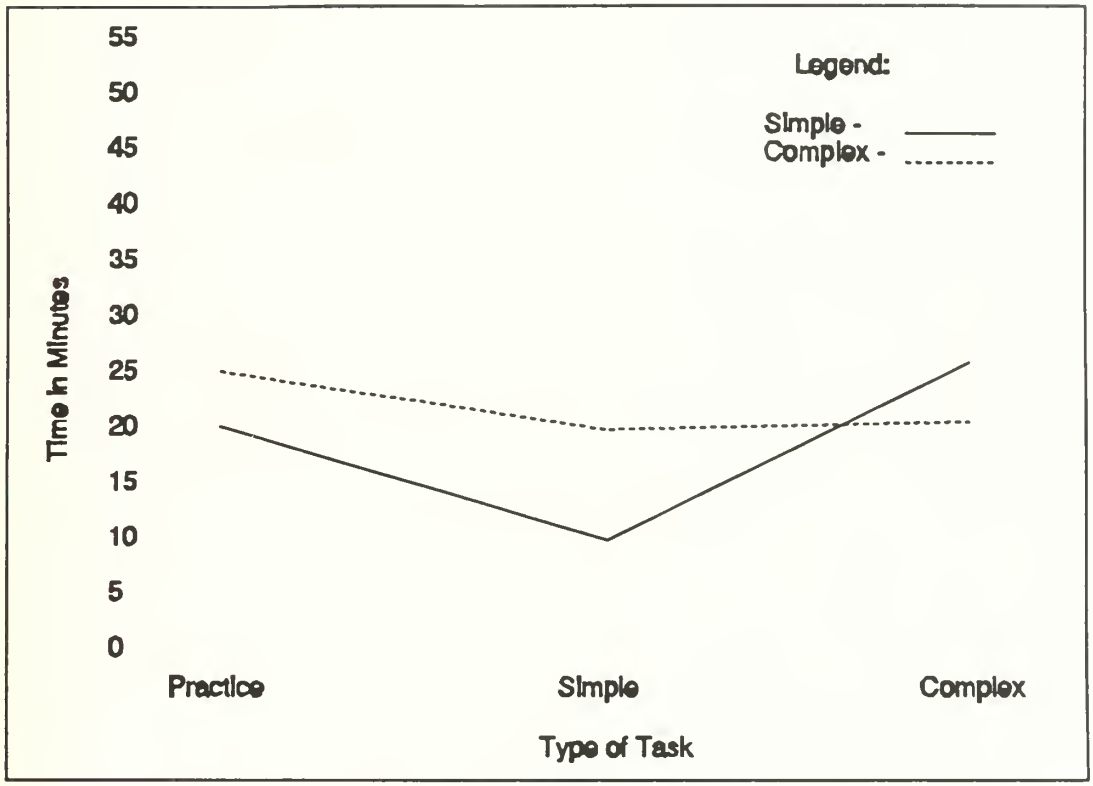

Figure 26

Mean Values for Task Completion Time, as a function of Interface Type and Task Type

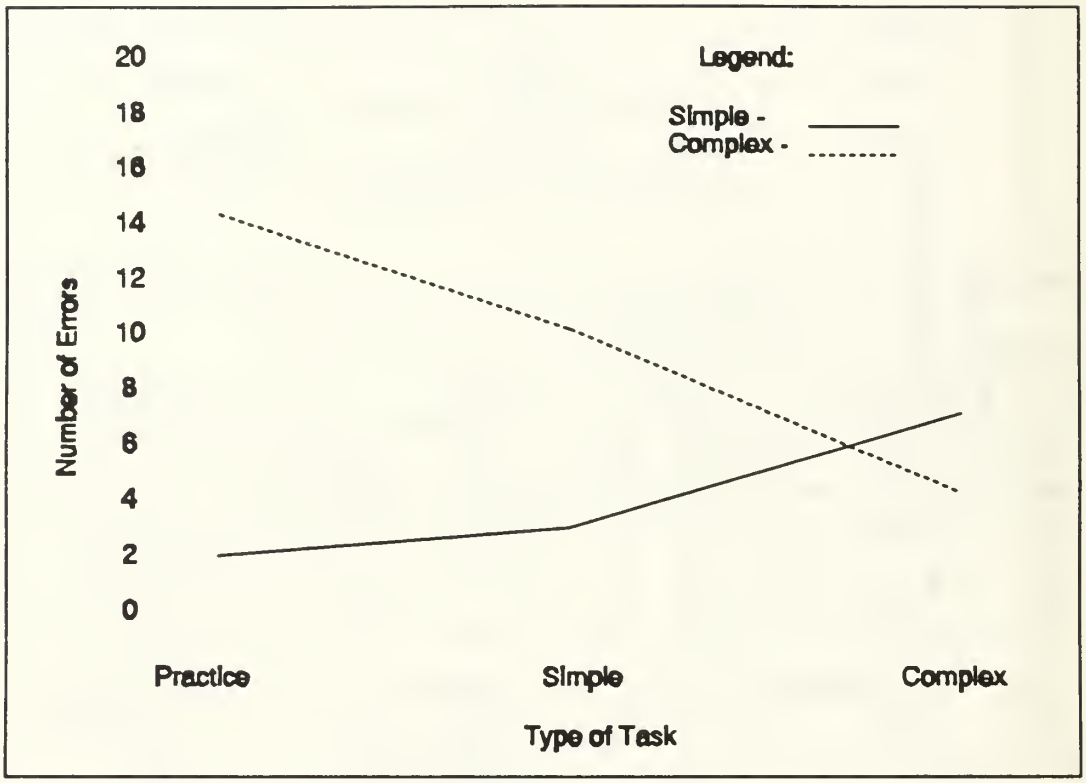

Figure 27

Mean Values for Number of Errors, as a Function of Interface Type and Task Type

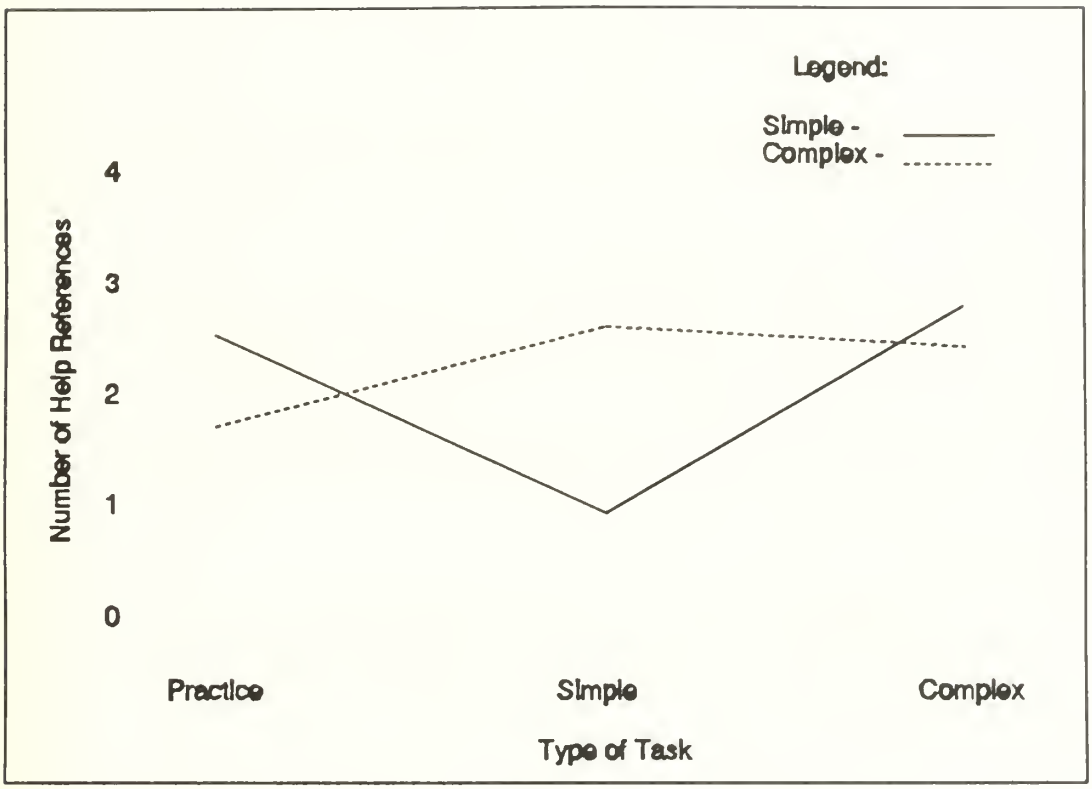

Figure 28

Mean Value of Number of Help References, as a Function of Interface Type and Task Type

1. Task Completion Time

Table 3 gives the results of a MANOVA for the dependent variable completion time.

Table 3
MANOVA FOR THE DEPENDENT VARIABLE COMPLETION TIME

	Source of Variation	d.f.	F	p
Between Subjects	Interface	1	3.15	0.0872
	Subjects w/in Cells	27		
Within Subjects	Task	2, 26	7.4228	0.0028
	Task x Interface	2, 26	6.1049	0.0067

A significant difference is shown in task completion time as a result of the between subjects variable of interface type (complex or simple). There was also a significant difference observed as a result of the within subjects variables of task and the interaction of the task and the interface.

The results of the univariate test (Table 4) show that the time required to learn the complex interface was not significantly different from the time required to learn the simple interface. This does not support hypothesis H_{1A}, that the time required to learn the complex interface would be greater than that required to learn the simple one.

Table 5 shows that the time required to complete the simple task set was significantly different for the complex interface than it was for the simple one. This does not support hypothesis H_{1B}, that there would be no significant difference in the task completion times for the two interfaces in the simple task set. Table 6 shows that the

Table 4
UNIVARIATE ANOVA FOR COMPLETION TIME FOR PRACTICE SESSIONS

Source of Variation	d.f.	F	p
Type of Interface	1	1.83	0.1871
Subjects within Cells	29		

Table 5
UNIVARIATE ANOVA FOR COMPLETION TIME FOR SIMPLE TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	11.07	0.0025
Subjects within Cells	29		

time required to complete the complex task set was significantly different for the simple interface than it was for the complex one. This supports hypothesis H_{1C}.

Table 6
UNIVARIATE ANOVA FOR COMPLETION TIME FOR COMPLEX TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	3.30	0.0805
Subjects within Cells	29		

Table 6
MANOVA FOR THE DEPENDENT VARIABLE NUMBER OF ERRORS

	Source of Variation	d.f.	F	p
Between Subjects	Interface	1	9.96	0.0039
	Subjects w/in Group	27		
Within Subjects	Task	2, 26	2.0120	0.1540
	Task x Interface	2, 26	16.0105	0.0001

2. Number of Errors

Table 7 is a summary of the results of a multivariate analysis of variance for the dependent variable number of errors. Performance results for the two interfaces were significantly different between subjects as a result of interface complexity. There was no significant performance difference looking within subjects as a function of task, but a significant difference was shown looking within subjects as a function of task and interface interaction.

Univariate analysis of variance tests were also conducted on the results for the number of errors variable, and confirmed using Scheffe's test. The number of errors that occurred during the practice session were significantly different for the subjects using the complex interface than for those using the simple interface, as seen in table 8. This supports hypothesis H_{2A}.

Table 9 shows there was a significant difference in the number of errors in the simple task set committed by subjects using the complex interface than there were errors committed by subjects using the simple interface. This does not support hypothesis H_{2B}, which predicted no significant difference in the number of errors in the simple task set.

Table 8
UNIVARIATE ANOVA FOR NUMBER OF ERRORS, PRACTICE SESSION

Source of Variation	d.f.	F	p
Type of Interface	2	29.74	0.0001
Subjects within Cells	29		

Table 10 shows no significant difference in errors occurring during the complex task set. This does not support hypothesis H_{2C} which was that there would be significantly more errors committed on the simple interface than the complex one.

Table 9
UNIVARIATE ANOVA FOR NUMBER OF ERRORS, SIMPLE TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	6.03	0.0208
Subjects within Cells	29		

3. Help References

Table 11 is a summary of the results of a multivariate analysis of variance for the dependent variable number of help references. Results show there were no significant differences when looking at the between subjects variable of interface complexity, or when looking at the within subjects variable of task complexity. There was a significant difference in the within subjects interaction of task complexity with interface complexity.

Table 10
UNIVARIATE ANOVA FOR NUMBER OF ERRORS, COMPLEX TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	1.72	0.2011
Subjects within Cells	29		

Table 11
MANOVA FOR THE NUMBER OF HELP REFERENCES

	Source of Variation	d.f.	F	p
Between Subjects	Interface	1	0.04	0.8492
	Subjects w/in Group	27		
Within Subjects	Task	2, 26	0.7431	0.4855
	Task x Interface	2, 26	6.8282	0.0041

Table 12 shows the univariate analysis of variance results for the number of times that help was referenced during the practice session. There was no significant difference in the number of times there was reference to help between the interfaces. This does not support hypothesis H_{3A}.

Table 12
UNIVARIATE ANOVA FOR NUMBER OF HELP REFERENCES, PRACTICE SESSION

Source of Variation	d.f.	F	p
Type of Interface	1	1.95	0.1738
Subjects within Cells	29		

Table 13 shows the univariate analysis results for the number of times help was referenced during the simple task set. The number of references to help was significantly greater for those using the complex task set than for those using the simple task set. This does not support hypothesis H_{3B} which was there would be no significant difference as a function of the interface used.

Table 13
UNIVARIATE ANOVA FOR NUMBER OF HELP REFERENCES, SIMPLE TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	5.40	0.0278
Subjects within Cells	29		

Table 14 shows the results of analysis of the number of times that help was referenced during the complex task set. There was no significance in the difference between the two interfaces. This does not support hypothesis H_{3C} which was there would be more help references for the simple interface.

Table 14
UNIVARIATE ANOVA FOR NUMBER OF HELP REFERENCES, COMPLEX TASK SET

Source of Variation	d.f.	F	p
Type of Interface	1	0.28	0.6009
Subjects within Cells	29		

Figure 15 is a summary of the test results produced using Scheffe's test. The means were significantly different for all three variables for the simple task set. The only variable about which groups were significantly different for the practice set was the number of errors variable, and there were no significant differences in any of the variables for the complex task set. For the Scheffe test alpha was 0.05 and the degrees of freedom were 27. The critical value of F was compared. The same results were obtained using Tukey's test. For that reason, the results are not presented here.

Table 15
SUMMARY OF SCHEFFE'S TEST

Dependent Variable	Practice	Simple	Complex
Completion Time	Not Significant	Significant	Not Significant
Number of Errors	Significant	Significant	Not Significant
Number of Help References	Not Significant	Significant	Not Significant

C. SUBJECT RESPONSES

Participants in the study were asked to respond to questions rating the use, ease of learning, helpfulness of error messages, and how rapidly progress was made with the interface used on a linear scale of 1 to 10. Results are provided in Figure 28. It can be seen that there was very little difference in the ease of use between the two interfaces, though the subjects found the simple interface somewhat easier to learn and there was more help gained from the error messages in the simple interface than the complex one. The subjects felt there was more rapid progress through the simple interface than through the complex one. The subjects were not comparing their experiences on one interface compared to the other, rather expressing an opinion as to how they felt about the particular interface they used.

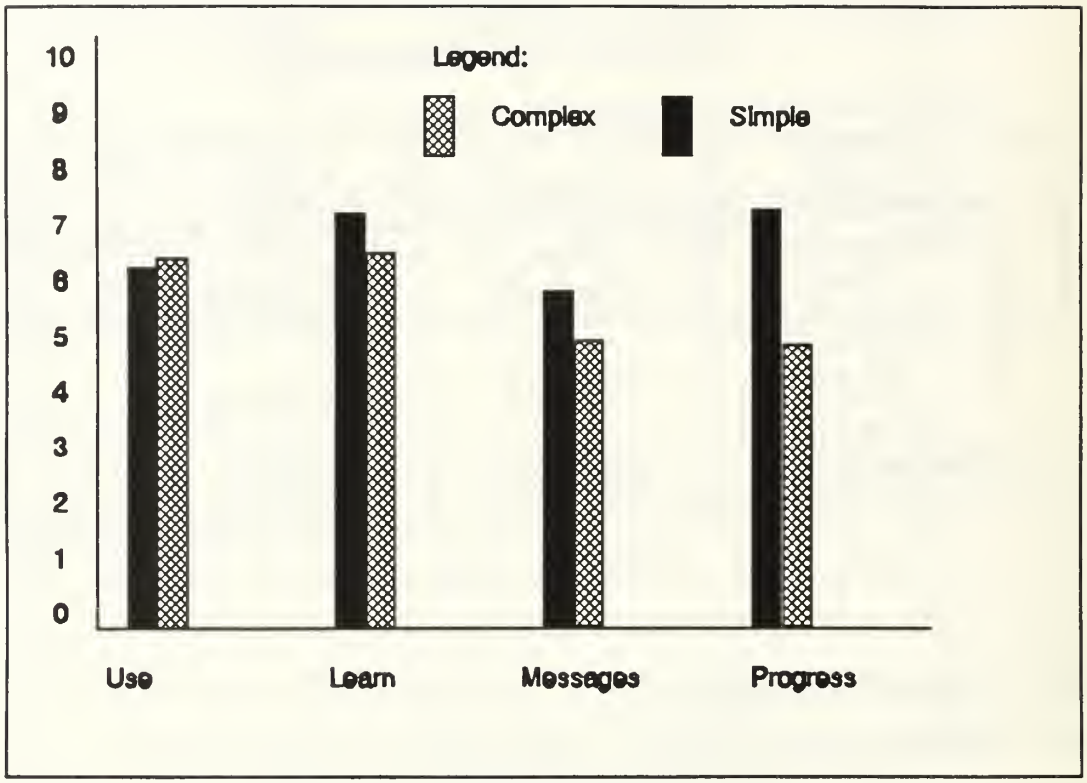

Figure 29
User Ratings of the Simple and Complex Interfaces for Four Variables

Table 16
RESULTS OF ANALYSIS OF VARIANCE FOR SURVEY QUESTIONS

df= 1, 28 Subjects Within Cells

Question	Interface	Mean	Standard Deviation	F	p
Ease of Use	Simple	4.7857	2.8871	0.09	0.7707
	Complex	6.2143	2.4862		
Ease of Learning	Simple	7.1429	1.6575	1.16	0.2915
	Complex	6.3571	2.1700		
Error Messages Helpful	Simple	5.7143	2.7296	0.76	0.3899
	Complex	4.7857	2.8871		
Rapid Progress	Simple	7.1429	1.5119	6.04	0.0210
	Complex	5.2143	2.5170		
VVQ	Simple	9.7143	2.4315	0.42	0.5242
	Complex	9.1429	2.2483		
Experience	Simple	8.4286	1.6968	6.70	0.0156
	Complex	6.6429	1.9457		

Table 16 shows the results of an analysis of variance for the four survey questions, as well as the results of the analysis of the Visualizer-Verbalizer Questionnaire, and the results obtained when the subjects were asked to rate their computer experience on a scale of one to ten. The p-value shows that there is a significance in the question asking how rapidly the subjects perceived that they progressed through the experiment. Further, there was a significant difference in the computer experience level. The results of the VVQ

show there was no significant difference between the subjects in the verbalizing/visualizing tendency.

The significant difference in the experience levels between the two interfaces was an unexpected result, and probably was the reason the hypotheses weren't more fully supported. The fact that the experience level of the subjects using the simple interface was so much higher than the experience level of the subjects using the complex interface certainly affected the outcome of the completion time and the number of errors committed. It may have also affected the number of help references if knowledge of interfaces used in the past could be transferred to the interfaces used in the experiment.

V. DISCUSSION AND CONCLUSIONS

A. GOAL

The goal of this study was to determine if the complexity of the interface had a significant effect on the computer user's productivity in the accomplishment of simple and complex tasks. To conduct the study, a group of 29 subjects was asked to perform a series of task sets, one a practice set, the second a simple task set and the third a complex task set. The group of subjects was split into two groups, one working on a simple direct manipulation interface and the second on a complex direct manipulation interface.

B. HYPOTHESES

The hypotheses were developed around the dependent variables of task completion time, number of errors committed, and the number of help references made by the subjects. They were supported or rejected using four statistical tests; multivariate analysis of variance, univariate analysis of variance, Scheffe's test, and Tukey's test. The hypotheses were that the completion times of the practice set would be significantly longer for the complex interface than for the simple interface (H_{1A}) and during the complex task set the completion times would be significantly longer for the simple interface than for the complex one (H_{1C}), but there would be no significant difference between the two for the simple task set (H_{1B}). There was no support for H_{1A} and H_{1B}. Hypothesis H_{1C} was supported. The second set of hypotheses was that there would be significantly more errors made on the complex interface than on the simple one during the practice set (H_{2A}) and significantly more errors made on the simple interface than on the complex one during the complex task set (H_{2C}), but there would be no significant difference during the simple task set (H_{2B}). Hypothesis H_{2A} was supported, but there was no support for hypotheses H_{2B} or H_{2C}. Although there was a difference observed in the data for the first two sets of hypotheses, and the difference was in the predicted direction, the difference did not reach statistical significance. The third set of hypotheses was that

there would be significantly more help references made by subjects using the complex interface than the simple one during the practice set (H_{3A}) and significantly more help references made by users of the simple interface during the complex task set (H_{3C}), but there would be no significant differences in the number of references between the two interfaces during the simple task set (H_{2B}). This set of hypotheses was not supported.

C. CONCLUSIONS OF STUDY

The study generally showed that it took a longer period of time to learn to use the complex interface, but as the task sets became more complex the completion time for the complex interface decreased. It took less time to complete the complex task set on the complex interface than it did on the simple one. While learning to use the complex interface, more errors were by the subjects than by those using the simple interface, but during the complex task set, more errors were made by the subjects using the simple interface. No useful conclusion could be drawn from the number of help references made by the subjects of either interface. The overall conclusion is that once the user has learned to use the complex interface, productivity improves as the level of complexity of the task increases past a certain point. The productivity of the user on the simple interface seems to decline as the task complexity increases past a certain point. Although the answers given to the survey questions were subjective, it appeared that the subjects perceived the complex interface to be easier to use, not much more difficult to learn, presented more helpful error messages and allowed them to proceed through the task sets more rapidly.

The model of the user's representation of a task in device dependent and device independent components is a beneficial one. That correlation can be illustrated in this experiment by the difference in results of the subjects accomplishing the same sets of tasks on different interfaces. It appeared that the complexity of the device did not handicap the users in the accomplishment of tasks, rather the more complex interface showed better results for user productivity for the more complex task set. The capability to predict the effort required of a subject to learn an interface and become productive with

it, as offered by the Kieras and Polson approach of user complexity, was also supported by the results.

D. POSSIBILITIES FOR FURTHER RESEARCH

Several follow on studies are suggested as a result of this study. First, several layers of complexity should be used to demonstrate at what level of complexity the improvement in performance is noted. Second, several levels of interface complexity should be introduced to show what level provides an ideal compromise between ease of learning and productivity. Third, an interface that combines the features of the direct manipulation interface, the command line interface, and the menu type interface could be studied to ascertain the most ideal mix of functionality. Fourth, the task set should be designed as a representative of a specific application, such as text editing or spreadsheet operations, and the subjects recruited should be familiar with the type of tasks presented, such as clerical personnel. Finally, the population of the test group should be large, so that more meaningful data can be obtained.

APPENDIX A

DEFINITIONS

Cognitive Complexity - Complexity dependent on the amount, content and structure of the knowledge required to operate a device or system (Kieras and Polson, 1985:pp. 364-365).

Cognitive Complexity Approach - Work based on the hypothesis that content, structure, and amount of knowledge required to perform a task on a system determines training time and productivity (Rasmussen and Zunde, 1987:p. 366).

Cognitive System - The human system that provides the ability to move symbolic information obtained from the sensory image stores to working memory, where it is combined with information previously stored in long-term memory (Card, Moran and Newell, 1983:p. 24).

Complexity - The number of goals and processes that must be controlled, and the characteristics of the means available for this control (Karat, 1987:p. 24); the level of complexity is determined by the difficulty of acquiring the new knowledge necessary to operate a device successfully (Rassmussen and Zunde, 1987:p. 366).

Declarative Knowledge - Knowledge of facts (Kieras and Polson, 1985:p. 369).

Device Complexity - The level of complexity of the knowledge representations required to operate a given device (Rassmussen and Zunde, 1987:p. 367).

Device Representation - The knowledge that a user has about the device itself (Rasmussen and Zunde, 1987:p. 366).

Direct Manipulation Interface - An interface that presents a set of visual representations on a display and provides a repertoire of manipulations that can be performed on any of them (Reinhard, 1991:p. 78).

Environment - Externally imposed requirements that a person must satisfy, and the equipment available for use in meeting these requirements.

GOMS Model - A representation of a user's task. GOMS is an acronym for Goals, Operators, Methods and Selection Rules.

Icons - Graphic representations of objects or operations.

Interface - The dialogue that allows communication between the human and the computer, the purpose of such dialogue being the accomplishment of a task (Card, Moran and Newell, 1983:p. 23).

Job Environment - A collection of task environments.

Operations (GOMS) - Mental representations of the various elementary functions that a device can perform and of other cognitive operations that occur during the execution of a task.

Production Rule - A statement about the external environment or the contents of working memory (Bovair, Kieras and Polson, 1990:p. 7).

Production System - A formal notation used to represent the user's knowledge of the job-task environment, which is composed of production rules and working memory (Rasmussen and Zunde, 1987:p. 368).

Task Complexity - The measurable complexity of the task that is being performed (Wood, 1987:p. 66).

Task Representation - A user's knowledge of how to carry out a using a given device (Rasmussen and Zunde, 1987:p. 366).

User Interface Complexity - The complexity of a device or system from the point of view of the user (Rasmussen and Zunde, 1987:p. 365).

Working Memory - The part of the memory that contains representations of current goals and inputs from the environment and other information about the state of current and past actions (Bovair, Kieras and Polson, 1990:p. 6).

APPENDIX B

SIMPLE DIRECT MANIPULATION INTERFACE

YOUR NAME: _____

SMC NO: _____

I. INTRODUCTION

The exercise in which you are about to participate involves operating and evaluating a simple direct manipulation operating system interface[5]. The functions you will be evaluating are the file manipulation features of this interface. You will be asked to read a short description of file management, followed by instructions for each file management operation. You will then be asked to practice each operation until you can perform the operation successfully and you feel comfortable about it. Then you will be asked to perform a series of tasks using the operations you have learned.

Privacy Act

The information accompanying this experiment will be used for data collection and correlation purposes only. Information provided is voluntary.

---

[5] This interface was developed for the original experiment by N. Reinhard using Smalltalk/V, a product of Digitalk, Inc. (Reinhard, 1991)

II. QUESTIONNAIRE

INSTRUCTIONS:

Please place a (T)true or (F)false next to each of the following statements as you feel it best applies to you.

___ 1. I enjoy doing work that requires the use of words.

___ 2. My daydreams are sometimes so vivid I feel as though I actually experience the scene.

___ 3. I enjoy learning new words.

___ 4. I can easily think of synonyms for words.

___ 5. My powers of imagination are higher than average.

___ 6. I seldom dream.

___ 7. I read rather slowly.

___ 8. I cannot generate a mental picture of a friend's face when I close my eyes.

___ 9. I don't believe that anyone can think in terms of mental pictures.

___ 10. I prefer to read instructions about how to do something rather than have someone show me.

___ 11. My dreams are sometimes vivid.

___ 12. I have better than average fluency in using words.

___ 13. My daydreams are rather indistinct and hazy.

___ 14. I spend very little time attempting to increase my vocabulary.

___ 15. My thinking often consists of mental pictures or images.

III. FILE MANAGEMENT SYSTEM

A. OPERATING SYSTEM

An operating system is the software program that makes the hardware useable. The operating system can accomplish many functions, among which are communicating between the user and the computer (user interface), allowing the sharing of hardware components among a number of users, managing memory assets, providing security, allowing the sharing of data among users and many other functions.

One of the primary duties of the operating system is to manage **files**. A short discussion of the organization and structure of files follows. If you are familiar with this topic, you may skip it, but it will only take a short time to read it.

B. DIRECTORIES AND FILES

A **software program** consists of one to several files. These files work together to make the application that the user sees and interacts with on the computer display device. A **directory** contains the files used by a program. A directory tree is a collection of all the directories contained in a given memory storage device. The directory tree resembles a tree that is upside down, with the root at the top and the branches at the bottom. The first directory in this structure is called the root directory, and it contains the table of contents for the layer of directories below it as well as certain special files the operating system uses to start and configure the computer for operation.

It is also possible for directories to have any number of **subdirectories**, which are individual directories contained within a parent directory. This may sound confusing, but it is really a very simple way to efficiently organize a disk system. For example, under the root directory you may have directories called \SOFTWARE, \BATCH, \MSDOS, \UTIL, \PROJECTS and \GAMES. Under the \SOFTWARE directory you may have separate subdirectories for a spreadsheet application, a database management system, a graphics program and a word processing program. In the \UTIL directory you may have a lot of individual utility routines as well as individual subdirectories that contain specific types of utilities. This type of directory system lends itself to an orderly organization of files.

Figure 1 shows the relationship between the directories, subdirectories and files in a typical directory tree. Notice that the top level directory is called the root directory. It is generally represented by a "\". The description of where a file or directory is located in the directory tree structure is called a **path**, and each path starts out with a "\", signifying that the path begins with the root directory.

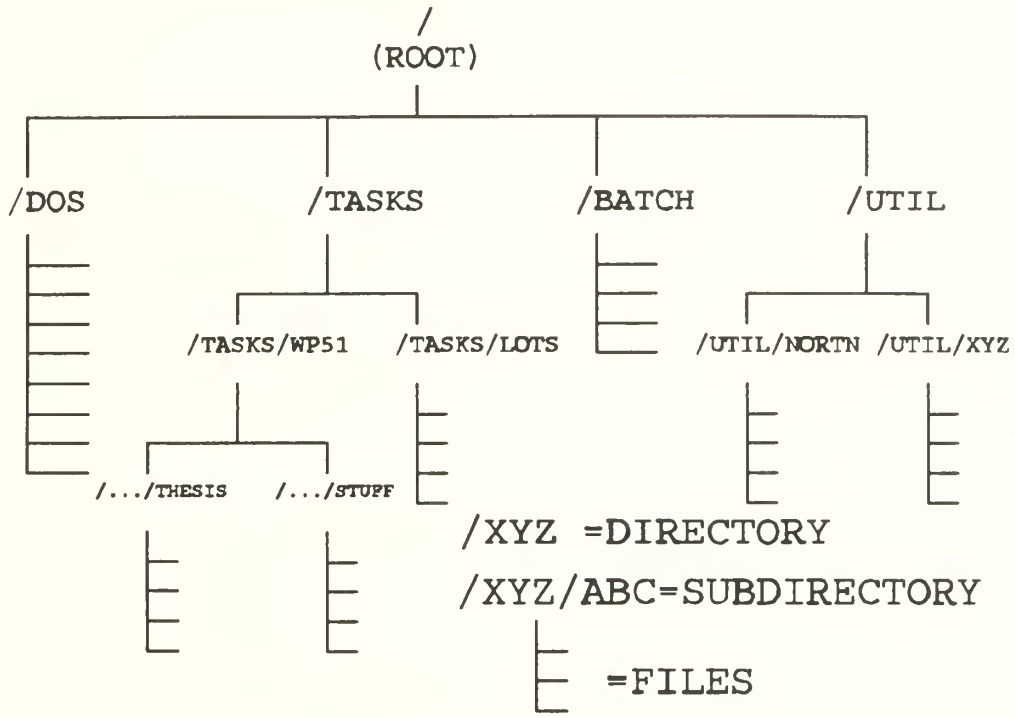

Figure 1
Relationships of Directories and Files in a Typical Directory Tree

Sometimes it is necessary to copy or move files from one directory to another. The operating system must provide the capability to accomplish this task. If, for example, you had produced a report on the word processor and stored it in the \..\STUFF subdirectory and wanted to use it in a thesis, which was stored in the \..\THESIS subdirectory, you would need some way to move the file from one subdirectory to the

other. Some form of the *copy* or *move* command would allow you to get this file into the desired directory. If you wanted to change the name of the file, the operating system must allow that capability.

In addition to manipulating the files within and between directories, the operating system must also provide a means to manage the directory system itself. Operations such as *creating* and *deleting* directories are needed for this function.

IV. OPERATING SYSTEM INTERFACE

The experimental operating system interface you will be working with is called a **Direct Manipulation Interface (DMI)**. It uses a pointing device, in this case a mouse, to select a graphical representation of an object or an operation, called an **icon**, or to select directory names or file names. Once selected, the icon carries out a specific function or operation, such as copying a file from one directory to another.

A. MOUSE

The mouse is shown in Figure 2. It is the pointing device used in this DMI and is the only input device you will use in this experiment. The keyboard will be used only for entering your name and SMC number at the beginning of the experiment.

Hold the mouse in the right hand so that the cord and buttons are at the top. Place your index and middle fingers lightly over the mouse buttons. Gently guide the movement of the mouse with the hand.

Normally the mouse is represented as an arrow on the screen, which moves as you move the mouse. When the system is performing an operation that takes longer than one second, the cursor will appear as an hourglass to indicate that the system is busy and you will have to wait to begin the next operation. The mouse is not functional when the hour glass is displayed on the screen (you may move the hour glass, but it is not possible to select an object).

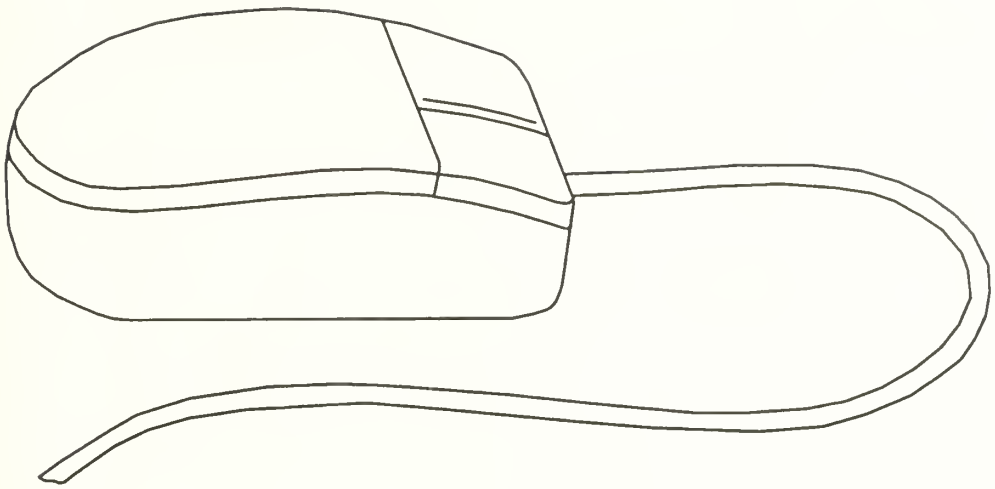

Figure 2
Mouse Control Device

Gently press or "click" the **left** button to select the object on the screen that the arrow or cursor is pointing to. Click the right button to call a menu in a given window that describes the operations you have an option of performing. Each window has a menu assigned to it, though most of the menus are not operational. The **help, sort, tree,** and **error** windows do have operational menus.

B. WINDOWS

The direct manipulation interface consists of five windows, which are shown in Figure 3. These windows are called (1) the Directory Window, (2) the File Window, (3) the File Sort Window, (4) the New Name Window, and (5) the Icon Window. Each window has a specific function and interacts with the other windows by use of mouse operations. Additionally, the Help and the Tree windows will pop up onto the screen when the icon that represents one of them is selected.

operations. Additionally, the Help and the Tree windows will pop up onto the screen when the icon that represents one of them is selected.

The characteristics of all the windows are essentially the same. The **Top Pane** is the main window and it includes all the other windows and fills the entire display screen. The **label** of the top pane contains the name of the interface or the name of a selected directory. The **Directory, File,** and **New Name Windows** all operate in a similar manner when the user selects an item. The selected item will appear highlighted. Due to the limited size of the windows, not all files, directories, or new names may fit in the window at one time. The window operations provide the ability to **scroll** the window. Scroll a window by pressing and **holding** the right mouse button (cursor changes to a four directional arrow) in the window you wish to scroll. Move the cursor out of the window in the direction you wish to scroll (keeping the mouse button depressed). The **scroll bar** on the right hand side of the window shows the status of the scrolling. The scroll bar is not visible unless a scrolling operation is in progress.

1. Directory Window

The Directory Window contains a list of all the directories on the disk in alphabetical order. The directories are indented to reflect their hierarchical or tree structure. For example, each subdirectory will be indented one space from the parent directory.

2. File Window

The name of the selected directory appears in reverse video (black background, white lettering) in the Directory Window and the names of all files contained in the directory are displayed in the File Window. Files listed in the File Window are in alphabetical order. All operations on files will be carried out using the File Window.

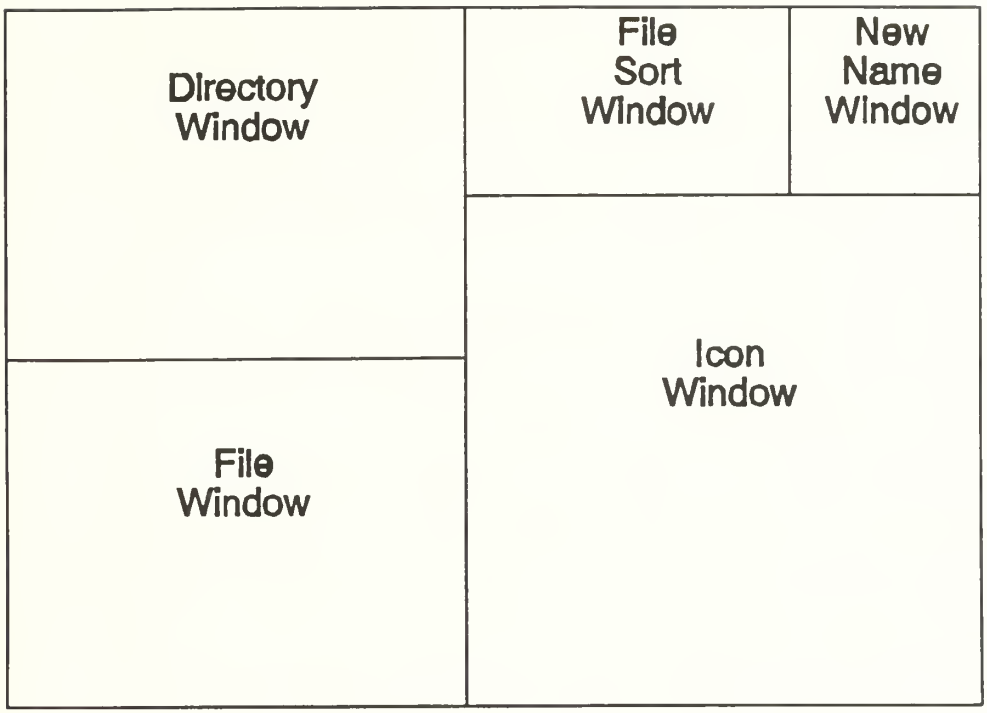

Figure 3
The Direct Manipulation Interface (DMI) Top Pane and Component Windows

3. The File Sort Window

The File Sort Window contains the names of all the files listed in the File window, plus additional information about these files. The File Sort Window has two parts, a label and a text portion. The label contains a menu, which allows the files to be sorted by name, date, or size. Items can be selected from this menu by clicking the right mouse button when the arrow is over the label and selecting with the left mouse button the desired menu option. The File Sort Window will then contain the files of the selected directory sorted by the specified method.

4. New Name Window

The New Name Window contains a list of all the names needed for new files, renamed files, and directories. The new name must be selected before an icon is selected to complete an operation on a file, if that operation involves naming the file. The system removes the deleted new name after it has been used.

5. Icon Window

The Icon Window is divided into three sections: File Icons, Directory Icons, and Other Icons. Using this window, files and directories can be created, copied, renamed, and deleted, as discussed later.

a. Create File Icon. The **Create File**icon creates a new file in the selected directory using the name selected in the New Name Window.

b. Copy File Icon. The **Copy File** icon copies a selected file to the selected directory using the selected name in the New Name Window.

c. Rename File Icon. The **Rename File** icon renames a selected file to the selected name in the New Name Window.

d. Delete File Icon. The **Delete File Icon** deletes the selected file from the selected directory.

e. Create Directory Icon. The **Create Directory** icon creates a new subdirectory under the selected directory using the selected name in the New Name Window.

f. Directory Tree. The **Directory Tree** icon displays a graphical depiction of the directories on the drive.

g. Delete Directory. The **Delete Directory** icon deletes the selected directory from the disk. The selected directory **cannot** contain files or subdirectories if it is to be deleted. All files must be deleted from a directory or a subdirectory before the directory or subdirectory may be deleted. In addition to all files

in the directory, all subdirectories must be deleted before a parent directory may be deleted.

h. Help. The **Help** icon displays the Help Window described later.

6. Prompter Window

When performing a file or directory operation, a "Prompter Window" will appear. This window allows you to confirm or cancel the operation. To make this window appears, click the **right** mouse button while the arrow is in the white portion of the window (suggested file or directory name). This will call up the window. Then select the "accept" or "cancel" option with the **left** mouse button. A Prompter Window will also appear when you attempt to conduct an operation without having specified all the necessary information. The needed information will appear in the white portion of the window. Remove the Prompter Window by selecting "cancel" from the prompter menu.

7. Help Window

The Help Window provides information on window operations, icons, and window locations. The Help Window can be displayed by selecting the Help icon. **Hold down** the left mouse button and a prompt appears for the upper left corner of the Help Window. **Move** the pop-up window corner to to the upper left corner of the screen and release the mouse button. The lower right corner of the window will appear and can be repositioned with the mouse. Click the **left** mouse button when the size of the window is at least 5 inches square.

The help commands appear in alphabetical order. When a command is selected, a description of the command will be provided in the right pane (text pane) of the window. The command list and text pane can be scrolled in the manner discussed earlier: press the right mouse button and **drag** the cursor out of the pane in the direction of the unseen text.

8. Error Window

Error Windows can appear either as a Prompter Window or as a pop-up window. A Prompter Window will be a small window with a short message. Selecting an icon without all the necessary information specified will cause it to appear. It can be

removed by (1) selecting the **right** mouse button when the cursor is located in the white portion of the prompter (area of short message) to obtain the menu options and then (2) selecting the "cancel" option.

If you attempt an operation that the operating system does not allow, an Error Pop-up Window will appear in the middle of the screen. The label, located at the top, will contain the error message. The remaining text portion of the window may be confusing and it is not necessary for you to understand it. To remove the window, use the mouse's **left** button and cursor to select a point outside the window, **or** select the menu with the **right** mouse button while the cursor is in the window label and then select the "close" option with the **left**mouse button.

9. Directory Tree

A Directory Tree is helpful for seeing the entire layout of the directory and subdirectory structure. As in the Directory Window, a subdirectory will branch off from its parent directory. The Directory Tree Window can be displayed by selecting the *Tree* icon. **Hold down** the **left** mouse button until a prompt appears at the left corner of the Tree Window. **Move** the Pop-up Window corner to the upper left corner of the screen and release the mouse button. The lower right corner of the window will appear and can be repositioned with the mouse. Click the **left** mouse button when the size of the window is approximately 5 to 6 inches square.

V. YOUR TASK

Your task is to practice each operation specified on the next page. It is important for you to understand each operation, because the step by step procedures will not be available during the actual experiment. Use the online help and Directory Tree as necessary. For data collection purposes, it is important that you conduct the experiment without delay.

VI. PRACTICE

A. FILE OPERATIONS:

NOTE: REPEAT EACH OPERATION UNTIL IT IS SUCCESSFULLY COMPLETED AND YOU FEEL COMFORTABLE ABOUT THE OPERATION

1. HELP SCREEN
   a. Select the **Help** icon, holding the left mouse button down.
   b. Move the cursor to the upper left corner of the screen and release the mouse button.
   c. Move the cursor to the lower right portion of the screen and press the left mouse button.
   d. Select the **Create File** item for information.
   e. To close the Help Window, press the right mouse button on the window label.
   f. Select "close".

2. CREATE FILE
   a. Select the *animals* directory
   b. Select the *bobcat* name from the New Name Window
   c. Select the **Create File** icon.
   d. Select the Prompter menu by pressing the right mouse button when the pointer is in the white portion of the Prompter Window.
   e. Select "accept" from the Prompter Window Menu.

3. SORT FILES
   a. Select the *special* directory from the Directory Window.
   b. Select the menu for the Sort Window by pressing the right mouse button when the pointer is over the label for the Sort Window.
   c. Select **Size** from the pop-up menu.

4. DELETE FILES
    a. Select the *people* directory from the Directory Window.
    b. Select the *presnt1* file from the File Window.
    c. Select the **Delete File** icon.
    d. Select the Prompter menu by pressing the right mouse button while the pointer is in the white portion of the Prompter Window.
    e. Select "accept" from the Prompter Window menu.

5. COPY FILE
    a. Select the *animals* directory from the Directory Window.
    b. Select the *cat* file from the File Window.
    c. Select the *kitty* name from the New Name Window.
    d. Select the **Copy** icon.
    e. Select the Prompter menu by pressing the right mouse button while the pointer is in the white portion of the Prompter Window.
    f. Select "accept" from the Prompter Window menu.

6. RENAME FILE
    a. Select the *people* directory from the Directory Window.
    b. Select the *girl* file from the File Window.
    c. Select the *woman* name from the New Name Window.
    d. Select the **Rename File** icon.
    e. Select the Prompter menu by pressing the right mouse button while the pointer is in the white portion of the Prompter Window.
    f. Select "accept" from the Prompter Window Menu.

B. DIRECTORY OPERATIONS:

1. CREATE DIRECTORY
    a. Select the *animals* directory from the Directory Window.
    b. Select the new name *mammals* from the New Name Window.
    c. Select the **Create Directory** icon.
    d. Select the Prompter menu by pressing the right mouse button while the pointer is in the white portion of the Prompter Window.
    e. Select "accept" from the Prompter Window menu.

2. TREE

a. Select the **Tree** icon, hold down the left mouse button and move the pointer to the upper left corner of the screen and release the button.

b. When the lower right corner of the window appears, position it to the lower right portion of the screen and click the left mouse button.

c. To close the Directory Tree, press the right mouse button on the Window Label.

d. Select "close".

3. DELETE DIRECTORY

a. Select the *mammals* directory from the Directory Window (Note: The directory must be empty in order to be deleted. All files and subdirectories must be deleted first.)

c. Select the **Delete Directory** icon.

d. Select the Prompter menu by pressing the right mouse button while the pointer is in the white portion of the Prompter Window.

e. Select "accept" from the Prompter Window menu.

****** STOP ******

***** CALL THE EXPERIMENTER TO PROCEED *****

VII. SIMPLE DIRECT MANIPULATION EXPERIMENT (SIMPLE TASK SET)

YOUR NAME: _____

SMC NO: _____

Complete the following operations using the procedures you have learned. Use the help screen as needed. Work at a normal pace and as accurately as possible.

1. Create a subdirectory of \ called *plots*.

2. Create a file called *twoplots.drs* in the *plots* directory.

3. Delete the file called *project.bak* in the *project* directory.

4. Copy the *plot.drs* file in the \ directory to *plot.bak* in the \ directory.

5. Rename the file called *twoplots.drs* in the *plots* directory to *twoplots.bak*.

**** STOP ****

*** CALL THE EXPERIMENTER BEFORE PROCEEDING ***

VII. SIMPLE DIRECT MANIPULATION INTERFACE (COMPLEX TASK SET)

Complete the following operations using the procedures you have learned. Use the help screen as needed. Work at a normal pace and as accurately as possible.

1. Copy the *package* file in the *business* directory to the *box* file in the *business* directory.

2. Rename the *papers* file in the *supplies* directory to *document* in the *supplies* directory.

3. Create a file called *car* in the *ground* directory and sort *ground* files by file size.

4. Delete the *planes* directory.

5. Find the largest file of **all** the directories and rename the file to *large.fil*.

6. Move the *west* subdirectory under the *flags* directory to the *transpor* directory.

IX. PLEASE COMPLETE THE FOLLOWING QUESTIONS

1. How easy was this interface to use?

(1 2 3 4 5 6 7 8 9 10)

not at all very much so

2. How easy was this interface to learn?

(1 2 3 4 5 6 7 8 9 10)

not at all very

3. To what extent were the error messages helpful?

(1 2 3 4 5 6 7 8 9 10)

not at all very

or N/A

4. How rapidly did you progress through this experiment?

(1 2 3 4 5 6 7 8 9 10)

not at all very

5. How close do you feel you were to 100% accuracy?

_____% (Consider errors in the simple and complex task but not in the practice task set)

Estimate the number of errors you made. _____

6. At any time during the experiment did the interface appear so difficult that you were ready to abandon the experiment?

YES NO

If so, at what point? _____

_____

X. PLEASE COMPLETE THE FOLLOWING QUESTIONS

1. Prior to this experiment what was your experience with computers?

(1	2	3	4	5	6	7	8	9	10)
none									much

2. If you have an experience level of 2 or greater from the previous question, from what type of operating system have you obtained the majority of your experience?

_____ MS DOS (or compatibles)

_____ UNIX

_____ Apple/Macintosh

_____ VAX

_____ Other (Please specify)

3. Complete the following using your knowledge of the operating system(s) specified above. Please specify the command if applicable.

a. How do you change directories, i.e., to the "WP51" directory?

_____

b. How do you list the files in the WP51 directory?

_____

c. How do you erase the "Miscellaneous" file from the WP51 directory?

_____

d. How do you find all batch files in the root directory?

_____

e. How do you copy all batch files to a floppy disk?

_____

f. How do you rename the *autoexec.bat* file to a backup file?

_____

g. How do you create a new *autoexec.bat* file?

_____

h. How do you changd a file to "read only"?

_____

i. How do you convert an *.exe* file to a *.com* file?

_____

4. What is your experience with using a mouse?

(1	2	3	4	5	6	7	8	9	10)
none									much

Please complete the following:

Age _____

Sex Male Female

Undergraduate Degree _____

NPS Curriculum _____

Years of computer experience _____

Thank you for your participation in this experiment.

APPENDIX C

COMPLEX DIRECT MANIPULATION INTERFACE

YOUR NAME: _____

SMC NO: _____

I. INTRODUCTION

The exercise in which you are about to participate involves operating and evaluating a commercial direct manipulation operating system interface[6]. The functions you will be evaluating are the file manipulation features of this interface. You will be asked to read a short description of file management, followed by instructions for each file management operation. You will then be asked to practice each operation until you can perform the it successfully and you feel comfortable about it. Then you will be asked to perform a series of tasks using the operations you have learned.

Privacy Act

The information accompanying this experiment will be used for data collection and correlation purposes only. Information provided is voluntary.

---

[6] The interface that will be used in this experiment is Microsoft's Windows 3.0.

II. QUESTIONNAIRE

INSTRUCTIONS:

Please place a (T)true or (F)false next to each of the following statements as you feel it best applies to you.

___ 1. I enjoy doing work that requires the use of words.

___ 2. My daydreams are sometimes so vivid I feel as though I actually experience the scene.

___ 3. I enjoy learning new words.

___ 4. I can easily think of synonyms for words.

___ 5. My powers of imagination are higher than average.

___ 6. I seldom dream.

___ 7. I read rather slowly.

___ 8. I cannot generate a mental picture of a friend's face when I close my eyes.

___ 9. I don't believe that anyone can think in terms of mental pictures.

___ 10. I prefer to read instructions about how to do something rather than have someone show me.

___ 11. My dreams are sometimes vivid.

___ 12. I have better than average fluency in using words.

___ 13. My daydreams are rather indistinct and hazy.

___ 14. I spend very little time attempting to increase my vocabulary.

___ 15. My thinking often consists of mental pictures or images.

III. FILE MANAGEMENT SYSTEM

A. OPERATING SYSTEM

An operating system is the software program that makes the hardware useable. The operating system can accomplish many functions, among which are communicating between the user and the computer (user interface), allowing the sharing of hardware components among a number of users, managing memory assets, providing security, allowing the sharing of data among users and many other functions.

One of the primary duties of the operating system is to manage **files**. A short discussion of the organization and structure of files follows. If you are familiar with this topic, you may skip it, but it will only take a short time to read it.

B. DIRECTORIES AND FILES

A **software program** consists of one to several files. These files work together to make the application that the user sees and interacts with on the computer display device. A **directory** contains the files used by a program. A directory tree is a collection of all the directories contained in a given memory storage device. The directory tree resembles a tree that is upside down, with the root at the top and the branches at the bottom. The first directory in this structure is called the root directory, and it contains the table of contents for the layer of directories below it as well as certain special files the operating system uses to start and configure the computer for operation.

It is also possible for directories to have any number of **subdirectories**, which are individual directories contained within a parent directory. This may sound confusing, but it is really a very simple way to efficiently organize a disk system. For example, under the root directory you may have directories called \SOFTWARE, \BATCH, \MSDOS, \UTIL, \PROJECTS and \GAMES. Under the \SOFTWARE directory you may have separate subdirectories for a spreadsheet application, a database management system, a graphics program and a word processing program. In the \UTIL directory you may have a lot of individual utility routines as well as individual subdirectories that contain specific types of utilities. This type of directory system lends itself to an orderly organization of files.

Figure 1 shows the relationship between the directories, subdirectories and files in a typical directory tree. Notice that the top level directory is called the root directory. It is generally represented by a "\". The description of where a file or directory is located in the directory tree structure is called a **path**, and each path starts out with a "\", signifying that the path begins with the root directory.

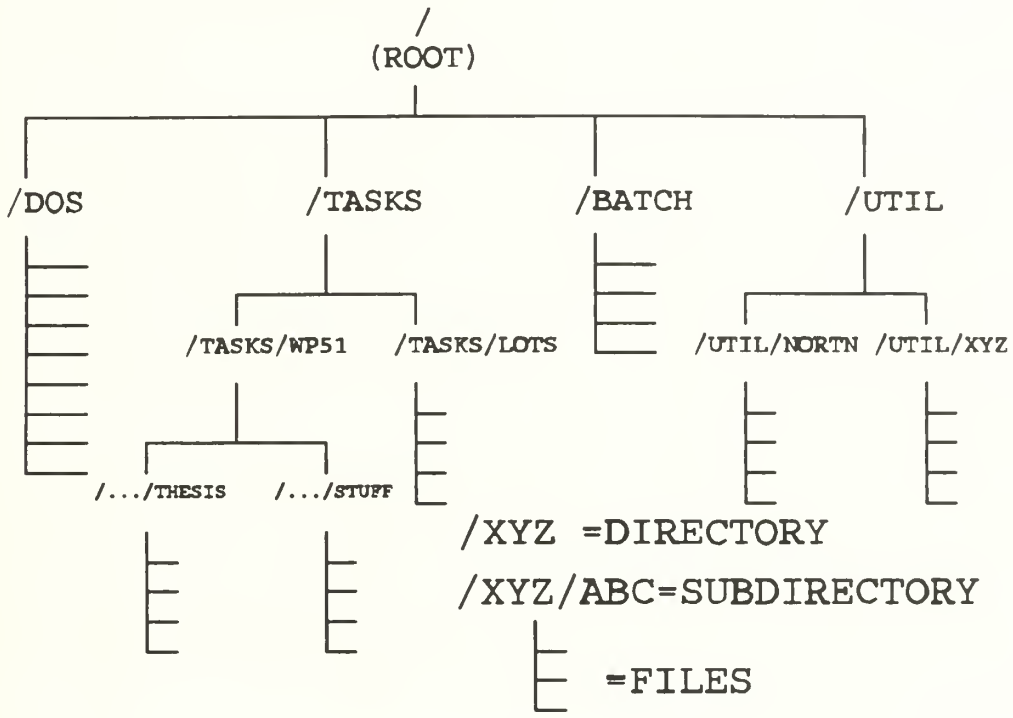

Figure 1
Relationships of Directories and Files in a Typical Directory Tree

Sometimes it is necessary to copy or move files from one directory to another. The operating system must provide the capability to accomplish this task. If, for example, you had produced a report on the word processor and stored it in the \..\STUFF subdirectory and wanted to use it in a thesis, which was stored in the \..\THESIS subdirectory, you would need some way to move the file from one subdirectory to the other. Some form

of the *copy* or *move* command would allow you to get this file into the desired directory. If you wanted to change the name of the file, the operating system must allow that capability.

In addition to manipulating the files within and between directories, the operating system must also provide a means to manage the directory system itself. Operations such as *creating* and *deleting* directories are needed for this function.

IV. OPERATING SYSTEM INTERFACE

The commercial operating system interface you will be working with is called a **Direct Manipulation Interface (DMI)**or, perhaps more popularly, a graphical user interface. It uses a pointing device, in this case a mouse, to select the graphical representation of an object or an operation, called an **icon**, or to select directory names or file names. Once selected, the icon carries out a specific function or operation, such as copying a file from one directory to another.

In this interface, you can access directories and view files in those directories through two objects, directory trees and directory windows. The initial screen you will see will be a directory tree of the disk drive that will contain the experiment directory structure. This view will show five options. They are File, Disk, Tree, View, Options, Window and Help. These options will be selected using the mouse.

A. MOUSE

The mouse is shown in Figure 2. It is the pointing device used in this DMI and will be the predominant input device you will use in this experiment, with supplementary inputs being performed on the keyboard.

Hold the mouse in the right hand so that the cord and buttons are at the top. Place your index and middle fingers lightly over the mouse buttons. Gently guide the movement of the mouse with the hand.

Normally the mouse is represented as an arrow on the screen, which moves as you move the mouse. When the system is performing an operation that takes longer than one second, the cursor will appear as an hourglass to indicate that the system is busy and you

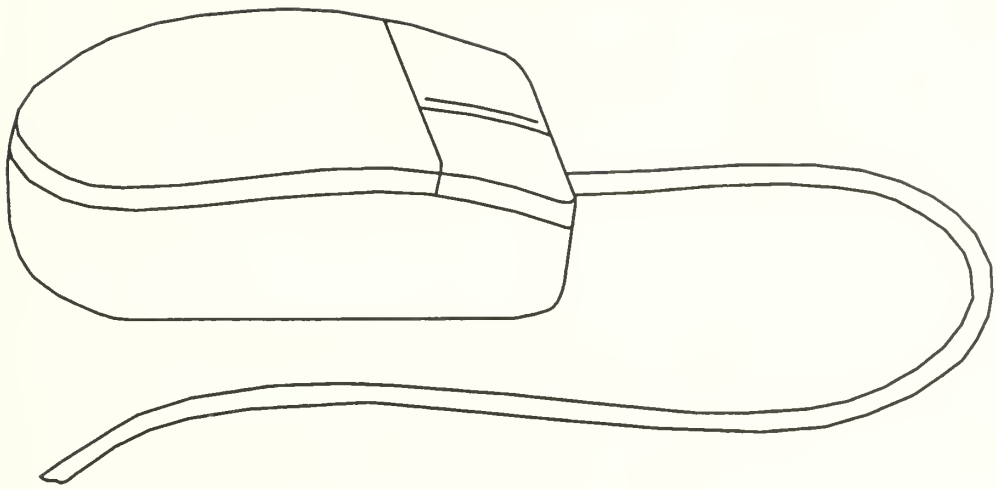

Figure 2
Mouse Control Device

will have to wait to begin the next operation. The mouse is not functional when the hour glass is displayed on the screen (you may move the hour glass, but it is not possible to select an object).

This interface can be run with only the left mouse button and five mouse control functions. Pointing involves moving the mouse on the desk top to move the diagonal arrow to the object of interest. Selecting or clicking is done with a single clicking of the left mouse button. Choosing or double clicking is also done with the left mouse button by depressing it twice. Dragging is done by moving the mouse with the left mouse button depressed. Marking is a click on a selection bar or dragging the mouse over a list or sections of a text by holding down the left button while moving the mouse.

B. WINDOWS

The screen can contain one or more windows. The window border is part of the window, even when the window takes up the entire screen, and serves as the window's boundary. The icon at the corner of the border is used to change the size of the window. The title bar contains the name of the window. The name of the application and document identifies the window. The Control menu appears when you click on the minus sign in the upper left hand corner of the window. The Control menu box call up *Restore, Move, Size, Minimize, Maximize, Close, and Switch To*. The Restore and Close functions are the ones you may find most use for. Close deactivates the window, making it disappear. Restore makes a window full size again after you have "minimized" it to an icon. The down arrow is the minimize box, which reduces the window to an icon. The maximize box is the up arrow, and makes the window fill the entire screen. When a window has been maximized, the box changes to a *Restore* function. The menu bar is used to choose application menus and to call up Help functions. The menus available depend on the application. The File menu contains such options as *New, Open, Save* and *Exit.* A pull-down menu drops to reveal options you can select. This menu usually appears directly below the menu item it belongs to. The horizontal and vertical scroll bars position the visible portions of objects that project horizontally of vertically beyond the area of the window. The highlighted squares on the scroll bars move along the bars and show the relative position of the visible portion of the object. When you begin each session, the screen will look like the screen in Figure 3.

C. FILE MANAGER

The interface application you will be working with is the File Manager. To get into File Manager, double click the icon. After a brief delay, the screen will be as that shown in Figure 4. This will be the initial mode the screen will show, with the appropriate disk drive selected. The disk drive the experiment directories will be on will be drive b:. There should be no reason for you to change this drive designation. Directory and file manipulation operations can be accomplished without leaving the File Manager.

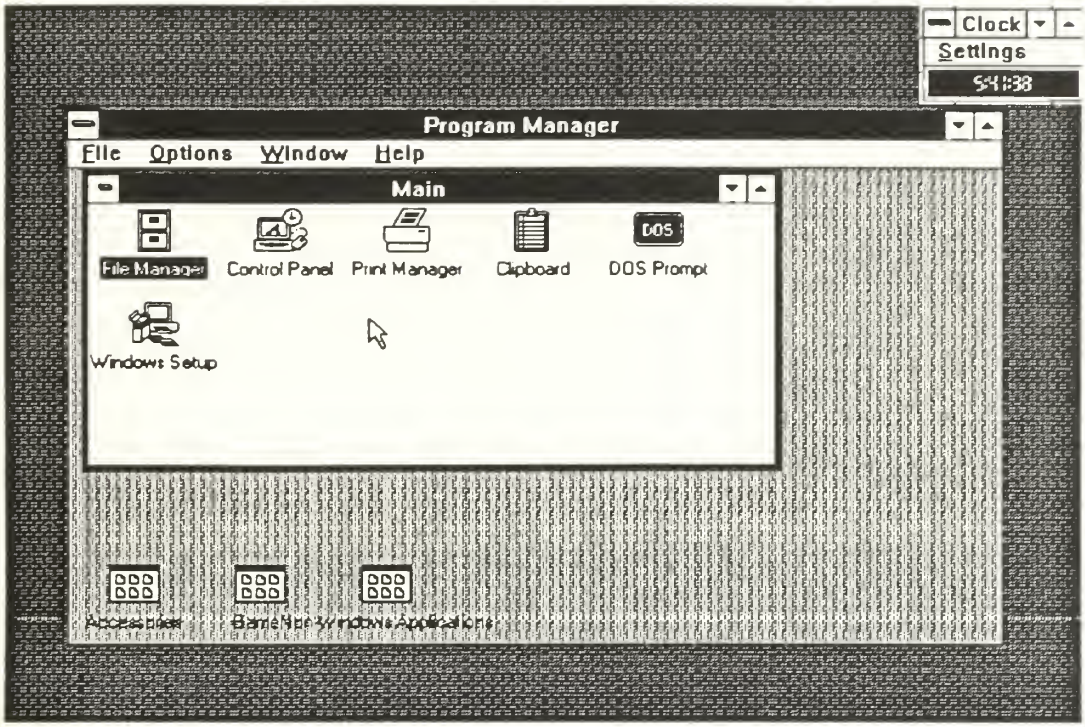

Figure 3
Opening Screen

The File Manager relies on the name of files and directories rather than on icons. A plus sign on a directory icon indicates the directory has subdirectories. Selecting the icon reveals the subdirectory name, and the directory icon takes on a minus sign. To see files within a directory you need only select its icon. An alphabetized list of files appears in the directories window. The View menu allows you to sort files by name, size, type or date.

When working with files or directories you highlight a file of directory and pull down the File menu. Then you can choose Create Directory within directories, or choose Rename, Copy, Move, or Delete commands for files and directories. Most of the commands bring up a Confirm dialogue box.

One command that is different is the one to create a file. In each of the experiment directory there is a directory called \NEWNAMES. The files in this directory have

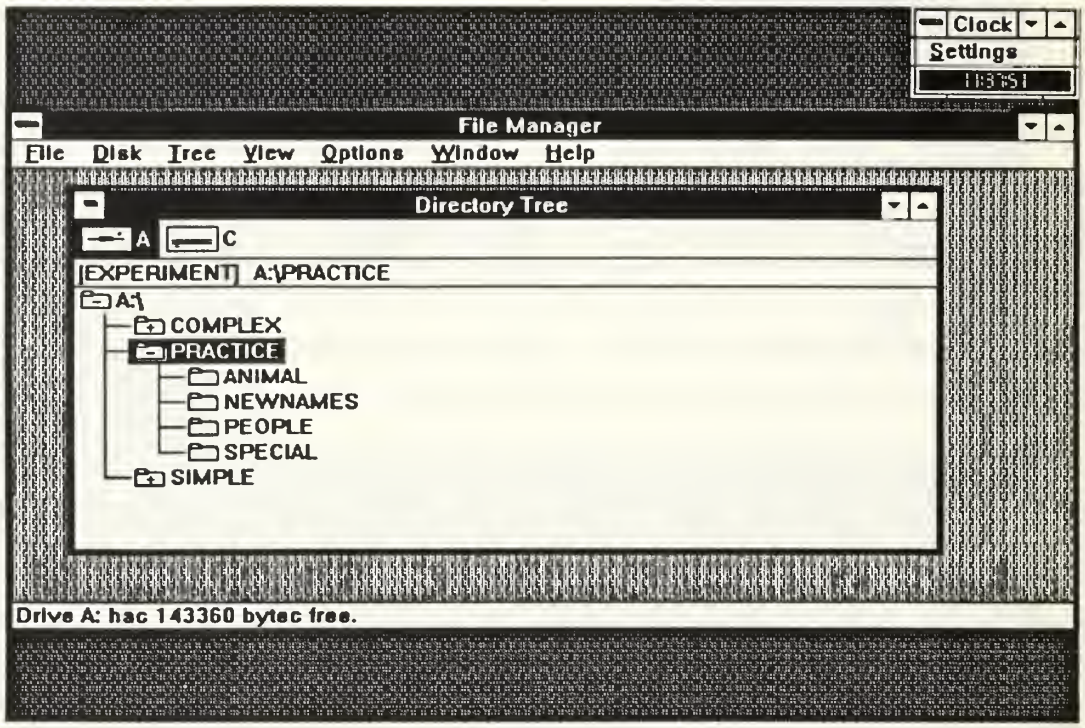

Figure 4
File Manager Screen

among them the names you will be asked to create files under. You will have to select these files and move them to the directory you have been asked to create the files under

D. HELP

The Help option brings up the Help window when it is selected. One of the choices in the window is the help index. This is the best selection to make to see the topics available to obtain information about.

E. ERRORS

If you make an error one of three things may occur. The first is nothing. If you attempted some action and there is no response when you thought there would be one, this is an error and should be counted as one. The second thing that may happen is that what you thought was going to occur didn't, rather some other event entirely transpired. This is also an error and should be so counted. The third thing that may occur is that an

error window will appear on the screen with information as to what you did that was incorrect.

V. YOUR TASK

Your task is to practice each operation specified on the next page. It is important for you to understand each operation, because the step by step procedures will not be available during the actual experiment. Use the online Help and Directory Tree as necessary.

You will notice blank spaces before and after each task set. This is for you to fill in the start times and end times for each set. The clock at the upper right hand corner of the screen is for your use. Also keep count of the number of incorrect actions taken (those that did not perform the intended function) and the number of times you accessed the online help function. Log these figures at the end of each task set. For data collection purposes, it is important that you conduct the experiment without delay.

VI. COMPLEX DIRECT MANIPULATION INTERFACE (PRACTICE SET)

A. FILE OPERATIONS:

NOTE: REPEAT EACH OPERATION UNTIL IT IS SUCCESSFULLY COMPLETED AND YOU FEEL COMFORTABLE ABOUT THE OPERATION

START TIME: _____

1. HELP SCREEN

   a. Select the **Help** option from the File Manager Window menu and click the left mouse button.
   b. Select **Index** from the **Help** menu that appears and click the left mouse button.
   c. Select the Down Scroll arrow and click the left mouse button until the **Command** section is in the window.
   d. Select the Tree Menu Command and click the left mouse button.
   e. Scroll through the Help text.
   f. Select the Index icon and click the left mouse button.
   g. Browse through the Help Window until you feel comfortable with it.
   h. Select the Control Menu from the Help Window (the minus sign in the upper left hand corner) and click the left mouse button.
   i. Select Close and click the left mouse button.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

2. CREATE FILE

   a. Select the *NEWNAMES* directory icon and double click the left mouse button.
   b. Select the *bobcat* file and click the left mouse button.
   c. Select File from the File Manager Window Menu and click the left mouse button.
   d. Select copy and click the left mouse button.
   e. In the window that appears, type "\animal.

f. Select copy and click the left mouse button.
g. Close the NEWNAMES Directory Window

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME:_____

3. SORT FILES

a. Select the *SPECIAL* directory from the Directory Tree Window and double click the left mouse button.
b. Select View from the File Manager Window Menu and click the left mouse button.
c. Select Sort by... and click the left mouse button.
d. Select size and click the left mouse button.
e. Select OK and click the left mouse button.
f. Close the SPECIAL directory window.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

4. DELETE FILES

a. Select the PEOPLE directory from the Directory Tree Window and double click the left mouse button.
b. Select the *presntl* file and click the left mouse button.
c. Select File from the File Manager Window Menu and click the left mouse button.

d. Select Delete... and click the left mouse button.
e. Select Delete and click the left mouse button.
f. Select Yes and click the left mouse button.
g. Close the PEOPLE Directory Window.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

5. COPY FILE

a. Select the ANIMAL directory from the Directory Tree Window and double click the left mouse button.
b. Select the *cat* file from the Directory Window and click the left mouse button.
c. Select File from the File Manager Window Menu and click the left mouse button.
d. Select Copy and click the left mouse button.
e. Type kitty.
f. Select Copy and click the left mouse button.
g. Close the ANIMAL Directory Window.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

6. RENAME FILE

a. Select the PEOPLE directory from the Directory Tree Window and double click the left mouse button.
b. Select the *girl* file from the Directory Window and click the left mouse button.

c. Select File from the File Manager Window menu and click the left mouse button.
d. Select Rename... and click the left mouse button.
e. Type woman.
f. Select Rename and click the left mouse button.
g. Close the PEOPLE Directory Window.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

B. DIRECTORY OPERATIONS:

START TIME: _____

1. CREATE DIRECTORY
   a. Select the ANIMAL directory from the Directory Tree Window and double click the left mouse button.
   b. Select File from the File Manager Window menu and click the left mouse button.
   c. Select Create Directory and click the left mouse button.
   d. Type mammals.
   e. Select OK and click the left mouse button.
   f. Select ANIMAL and click the left mouse button.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

2. MOVE DIRECTORY

   a. Select MAMMALS directory and double click the left mouse button.
   b. Select File from the File Manager Window menu and click the left mouse button.
   c. Select Move and click the left mouse button.
   d. Type "\".
   e. Select Move and click the left mouse button.
   f. Select Yes and click the left mouse button.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

3. DELETE DIRECTORY

   a. Select the MAMMALS directory from the Directory Tree Window and click the left mouse button. (Note: The directory must be empty in order to be deleted. All files and subdirectories must be deleted first.)
   c. Select File from the File Manager Window Menu and click the left mouse button.
   d. Select Delete and click the left mouse button.
   e. Select Yes and click the left mouse button.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

****** STOP ******

***** CALL THE EXPERIMENTER TO PROCEED *****

VII. COMPLEX DIRECT MANIPULATION EXPERIMENT (SIMPLE TASK SET)

YOUR NAME: _____

SMC NO: _____

Complete the following operations using the procedures you have learned. Use the help screen as needed. Work at a normal pace and as accurately as possible.

START TIME: _____

1. Create a subdirectory of \ called *plots*.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

2. Create a file called *twoplots.drs* in the *plots* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

3. Delete the file called *project.bak* in the *project* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

4. Copy the *plot.drs* file in the \ directory to *plot.bak* in the \ directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

5. Rename the file called *twoplots.drs* in the *plots* directory to *twoplots.bak.*

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

****** STOP ******

*** CALL THE EXPERIMENTER BEFORE PROCEEDING ***

VII. COMPLEX DIRECT MANIPULATION INTERFACE (COMPLEX TASK SET)

Complete the following operations using the procedures you have learned. Use the help screen as needed. Work at a normal pace and as accurately as possible.

START TIME: _____

1. Copy the *package* file in the *business* directory to the *box* file in the *business* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

2. Rename the *papers* file in the *supplies* directory to *document* in the *supplies* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

3. Sort the files in the *ground* directory by file size.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

4. Delete the *planes* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

START TIME: _____

5. Find the largest file of **all** the directories and rename the file to *largefil.*

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____**START TIME:** _____

___

6. Move the *west* subdirectory under the *flags* directory to the *transpor* directory.

STOP TIME: _____

NUMBER OF ERRORS: _____

NUMBER OF TIMES HELP ACCESSED: _____

IX. PLEASE COMPLETE THE FOLLOWING QUESTIONS

1. How easy was this interface to use?

(1 2 3 4 5 6 7 8 9 10)
not at all very much so

2. How easy was this interface to learn?

(1 2 3 4 5 6 7 8 9 10)
not at all very

3. To what extent were the error messages helpful?

(1 2 3 4 5 6 7 8 9 10)
not at all very
or N/A

4. How rapidly did you progress through this experiment?

(1 2 3 4 5 6 7 8 9 10)
not at all very

5. How close do you feel you were to 100% accuracy?
_____% (Consider errors in the simple and complex task but not in the practice task set)

Estimate the number of errors you made. _____

6. At any time during the experiment did the interface appear so difficult that you were ready to abandon the experiment?

YES NO

If so, at what point? _____

X. PLEASE COMPLETE THE FOLLOWING QUESTIONS

1. Prior to this experiment what was your experience with computers?

(1 2 3 4 5 6 7 8 9 10)
none much

2. If you have an experience level of 2 or greater from the previous question, from what type of operating system have you obtained the majority of your experience?

_____ MS DOS (or compatibles)

_____ UNIX

_____ Apple/Macintosh

_____ VAX

_____ Other (Please specify) _____

3. Complete the following using your knowledge of the operating system(s) specified above. Please specify the command if applicable.

   a. How do you change directories, i.e., to the "WP51" directory?

   _____

   b. How do you list the files in the WP51 directory?

   _____

   c. How do you erase the "Miscellaneous" file from the WP51 directory?

   _____

   d. How do you find all batch files in the root directory?

   _____

   e. How do you copy all batch files to a floppy disk?

   _____

   f. How do you rename the *autoexec.bat* file to a backup file?

   _____

g. How do you create a new *autoexec.bat* file?

_____

h. How do you change a file to "read only"?

_____

i. How do you convert an *.exe* file to a *.com* file?

_____

4. What is your experience with using a mouse?

(1	2	3	4	5	6	7	8	9	10)
none									much

Please complete the following:

Age _____

Sex Male Female

Undergraduate Degree _____

NPS Curriculum _____

Years of computer experience _____

Thank you for your participation in this experiment.

APPENDIX D

SAMPLE USER LOG

Sample, John
4444
Aug 16, 1991
DMI
SIMPLE

12:09:18	SDN	b:
12:09:35	SNN	plots
12:09:43	CD1	ERROR create dir without selectedDirectory
12:09:45	SDN	b:\
12:09:54	SDN	b:
12:09:54	SDN	b:\plots
12:09:55	CDS	Create directory b:\plots
12:10:02	SNN	twoplots.drs
12:10:07	CFS	Create New File b:\plots\twoplots.drs
12:10:08	SDN	b:\plots
12:10:11	HLP	Help window displayed
12:10:18	SHP	Delete File
12:10:29	SDN	b:\project
12:10:40	SFN	project.ins
12:10:47	DFS	Delete file \project\project.ins
12:10:47	SDN	b:\project
12:10:56	SFN	twoplots.drs
12:11:00	SNN	plot.bak
12:11:10	SDN	b:\project
12:11:12	PFS	Copy file b:\project\twoplots.drs to b:\project\plot.bak
12:11:14	SDN	b:\plots
12:11:19	SFN	twoplots.drs
12:11:26	SNN	twoplot.bak
12:11:33	RFS	Rename file b:\plots\twoplots.drs to b:\plots\twoplot.bak
12:11:41	DTS	tree displayed
12:11:57	WBW	WALKBACK window generated: directory not empty
12:12:13	SFN	twoplot.bak
12:12:28	DFS	Delete file \plots\twoplot.bak
12:12:28	SDN	b:\plots
12:12:35	DDS	Delete directory b:\plots\

APPENDIX E

CODE MEANINGS FOR SIMPLE INTERFACE USER LOG

CODE	MESSAGE
CD1	ERROR create directory without selected directory
CD2	ERROR create directory without selected name
CDS	Create directory successful
DD1	ERROR delete directory without selected directory
DD2	Delete directory successful
PF1	ERROR copy file without selected file
PF2	ERROR copy file without selected name
PFS	Copy file successful
CF1	ERROR create file without selected directory
CF2	ERROR create file without selected name
CFS	Create file successful
RF1	ERROR rename file without selected file
RF2	ERROR rename file without selected name
RFS	Rename file successful
DF1	ERROR delete file without selected file
DFS	Delete file successful
WBW	ERROR SmallTalk WalkBack window generated because of DOS error
SDN	Select directory name
SNN	Select new name
SFN	Select file name
SHP	Select help
HLP	Select help object
SPS	Sort files successful
DTS	Directory Tree Displayed

LIST OF REFERENCES

1. Bovair S., D. E. Kieras and P. G. Polson. "The Acquisition and Performance of Text-Editing Skill: A Cognitive Complexity Analysis." Human Computer Interaction 5, 1990, pp. 1-48. Lawrence Erlbaum Associates, Inc.

2. Campbell D. J. "Task Complexity: A Review and Analysis." Academy of Management Review 13(1), 1988, pp. 40-52.

3. Card S., T. P. Moran and A. Newell. The Pyschology of Human-Computer Interaction. Lawrence Erlbaum Associates, Publishers, 1983.

4. Karat J., R. Fowler and M. Gravelle. "Evaluating User Interface Complexity." Human Computer Interaction - INTERACT '87, 1987. Elsiever Science Publishers B.V.

5. Kieras D. E. "Problems is Using Cognitive Complexity Models." In M. Helander (Ed.), Handbook of Human Computer Interaction. Elsevier Science Publishers, B.V., 1988.

6. Kieras D. and P. G. Polson. "An approach to the formal analysis of user complexity." International Journal of Man-Machine Studies 22, 1985, pp. 365-394. Academic Press, Inc.

7. Margono S. and B. Shneiderman. "A Study of File Manipulation by Novices Using Commands vs. Direct Manipulation." 26th Annual Technical Symposium, Washington DC Chapter of ACM, 11/June 1987, pp. 154-159. Gaithersburg, MD: Association for Computing Machinery, Inc.

8. Newell A. and H. A. Simon. Human Problem Solving. Englewood Cliffs, New Jersey: Prentis-Hall, 1972.

9. Rasmussen J. and P. Zunde. Empirical Foundations of Information and Software Science III. Plenum Press, 1987.

10. Reinhard N. A. The Effect of Task Complexity on User Interfaces: A Comparison of Command Language Interface and Direct Manipulation Interface, Thesis. Monterey, California: Naval Postgraduate School, 1991.

11. Richardson A. "Verbalizer-Visualizer: A Cognitive Style Dimension." Journal of Mental Imagery 1(1), 1977.

12. Wood R. E. "Task Complexity: Definition of the Construct." Organizational Behavior and Human Decision Processes 37, 1986, pp. 60-82. . Academic Press, Inc.

BIBLIOGRAPHY

BEECH, D. ET. AL., "Concepts in User Interfaces: A Reference Model for Command and Response Languages," in *Lecture Notes in Computer Science*, vol. 234, Springer-Verlag, North Holland, 1986.

BOEHM, B. W., *Software Risk Management*, IEEE Computer Society Press, 1989.

BOOCH, G., "Object-Oriented Development," *IEEE Transactions on Software Engineering*, SE-12, 2, February (1986), 211-221.

BOVAIR, S., D. E. KIERAS AND P. G. POLSON, "The Acquisition and Performance of Text-Editing Skill: A Cognitive Complexity Analysis," *Human Computer Interaction*, 5 (1990), 1-48. Lawrence Erlbaum Associates, Inc.

CAMPBELL, D. J., "Task Complexity: A Review and Analysis," *Academy of Management Review*, 13, 1 (1988), 40-52.

CARD, S., T. P. MORAN AND A. NEWELL, *The Pyschology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Publishers, 1983.

CHECHILE, R. A., R. G. EGGLESTON AND R. N. FLEISCHMAN, "Modeling the Cognitive Content of Displays," *Human Factors*, 31(1) (1989), 31-43. The Human Factors Society, Inc.

FRYE, D. AND E. SOLOWAY, "Interface Design: A Neglected Issue in Educational Software," *CHI + GI 1987* (1987). Association for Computing Machinery.

GOODWIN, M., *User Interfaces in C++ and Object Oriented Programming*, Management Information Sources, Inc., Portland, OR, 1989.

HARRISON, M. AND H. THIMBLEBY, "The Role of Formal Methods in Human-Computer Interaction," in M. Harrison and H. Thimbleby (Ed.), *Formal Methods in Human-Computer Interaction*, Cambridge University Press, 1990.

HU, D., *Object-Oriented Environment in C++: A User-Friendly Interface*, Management Information Source, Inc, Portland, OR, 1990.

JACOB, R. J., "A Specification Language for Direct-Manipulation User Interfaces," *ACM Transactions on Graphics*, 5, 4, October (1986), 283-317.

JERRAMS-SMITH, J., "An attempt to incorporate expertise about users into an intelligent interface for Unix," *International Journal of Man-Machine Studies*, 31 (1989), 269-292. Academic Press Limited

KARAT, J., R. FOWLER AND M. GRAVELLE, "Evaluating User Interface Complexity," *Human Computer Interaction-INTERACT '87* (1987). Elsevier Science Publishers B.V.

KIERAS, D. AND P. G. POLSON, "An approach to the formal analysis of user complexity," *International Journal of Man-Machine Studies*, 22 (1985), 365-394. Academic Press, Inc.

MARGONO, S. AND B. SHNEIDERMAN, "A Study of File Manipulation by Novices Using Commands vs. Direct Manipulation," *26th Annual Technical Symposium, ACM* (1987). Association for Computing Machinery, Inc., Washington DC

MARK, W., "Knowledge-Based Interface Design," in D. Norman and S. Draper (Ed.), *User Centered System Design*, Lawrence Erlbaum Associates, Publishers, 1986.

MAULE, A. J., "Cognitive Approaches to Decision Making," in G. Wright (Ed.), *Behavioral Decision Making*, Plenum Press, 1985.

MCDONALD, J. E. AND SCHVANEVELDT ROGER W., "The Application of User Knowledge to Interface Design," in *Cognitive Science and its Applications for Human-Computer Interaction*, Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1988.

MIYATA, Y. AND D. NORMAN, "Psychological Issues in Support of Multiple Activities," in *User Centered System Design*, Lawrence Erlbaum Associates, Publishers, 1986.

NORMAN, D. A., "Cognitive Engineering," in D. Norman and S. Draper (Ed.), *User Centered System Design*, Lawrence Erlbaum Associates, Publishers, 1986.

PAYNE, J. W., "Psychology of Risky Decisions," in G. Wright (Ed.), *Behavioral Decision Making*, Plenum Press, 1985.

POLSON, P. G., "The Consequences of Consistent and Inconsistent User Interfaces," in *Cognitive Science and its Applications for Human-Computer Interaction*, Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1988.

RASMUSSEN, J. AND K. J. VICENTE, "Coping with human errors through system design: implications for ecological interface design," *International Journal of Man-Machine Studies*, 31 (1989), 517-534. Academic Press Limited.

REINHARD, N. A., *The Effect of Task Complexity on User Interfaces: A Comparison of Command Language Interface and Direct Manipulation Interface*, Thesis, Naval Postgraduate School, 1991.

RICHARDSON, A., "Verbalizer-Visualizer:A Cognitive Style Dimension," *Journal of Mental Imagery*, 1, 1, Spring (1977), 109-126. Brandon House, Inc.

RILEY, M. S., "User Understanding," in D. Norman and S. Draper (Ed.), *User Centered System Design*, Lawrence Erlbaum Associates, Publishers, 1986.

RIZZO, A., S. BAGNARA AND M. VISCIOLA, "Human error detection processes," *International Journal of Man-Machine Studies*, 27 (1987), 555-570. Academic Press, Limited.

ROGERS, Y., "Icons at the interface: their usefulness," *Interacting with Computers*, 1, 1 (1989), 105-117. , Butterworth & Co. (Publishers) Ltd.

RUBENSTEIN AND HERSH, *The Human Factor: Designing Computer Systems for People*. 1984.

SCHIELE, F. AND T. GREEN, "HCI Formalisms and Cognitive Psychology: The Case of Task-Action Grammar," in M. Harrison and H. Thimbleby (Ed.), *Formal Methods in Human-Computer Interaction*, Cambridge University Press, 1990.

SHNEIDERMAN, B., *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop Publishers, Inc., 1980.

SHNEIDERMAN, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishing Company, 1987.

TE'ENI, D., "Direct Manipulation as a Source of Cognitive Feedback: A human-computer experiment with a judgement task," *Working Paper 89-05* (July 1989). Case Western Reserve University, Cleveland, OH.

TE'ENI, D., "Perceived costs and benefits as determinants of user behaviour: an experiment with matchmaking," *Behaviour & Information Technology*, 9, 1 (1990), 31-45. Taylor & Francis, Ltd.

VAN HOE, R., K. POUPEYE, A. VANDIERENDONCK AND G. DE SOETE, "Some effects of menu characteristics and user personality on performance with menu-driven interfaces," *Behaviour & Information Technology*, 9, 1 (1990), 17-29. Taylor & Francis Ltd., Ghent, Belgium.

WASSERMAN, A. I., "Extending State Transition Diagrams for the Specification of Human-Computer Interaction," *IEEE Transactions in Software Engineering*, 11, 8 (1985), 561-575.

WEINBERG, G., *The Psychology of Computer Programming*. 1971.

WHITE, K., P. SHEEHAN AND R. ASHTON, "Imagery Assessment: A Survey of Self-Report Measures," *Journal of Mental Imagery*, 1, 1, Spring (1977), 145-170. Brandon House, Inc.

WILSON, J. R. R., ANDREW, "Mental Models: Theory and Application in Human Factors," *Human Factors*, 31, 6 (1989), 617-634. , The Human Factors Society, Inc, Nottingham, England.

WOOD, R. E., "Task Complexity: Definition of the Construct," *Organizational Behavior and Human Decision Processes*, 37 (1986), 60-82. Academic Press, Inc.

ZIEGLER, J., H. HOPPE AND FÄHNRICH, "Learning and Transfer for Text and Graphics Editing with a Direct Manipulation Interface," *CHI'86 Proceedings* (April 1986), 72-77.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	CDR John B. Frank, Jr, USN 7710 Killebrew Dr. Annandale, VA 22003	3
4.	Prof. Kishore Sengupta Code AS/SE Naval Postgraduate School Monterey, CA 93943	1
5.	Prof. Alice Crawford Code AS/CR Naval Postgraduate School Monterey CA 94934	1
6.	Prof. Tung Bui Code AS/BD Naval Postgraduate School Monterey, CA 93943	1
7.	CDR Tom Hoskins, USN Code 37 Naval Postgraduate School Monterey, CA 93943	1

Thesis
F78223 Frank
c.1 Designing the user in-
terface : considering the
concept of complexity.

Thesis
F78223 Frank
c.1 Designing the user in--
terface : considering the
concept of complexity.

3 2768 00034122 6