THE EFFECT OF INCONSISTENCIES IN A GRAPHICAL USER INTERFACE: OBJECTIVE AND SUBJECTIVE ASSESSMENTS

by

Richard H. Miller

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial & Systems Engineering

APPROVED:

_____

Robert J. Beaton, Chair

_____ _____

Paul T. Kemmerling, Jr. James Wilson

April, 1991

Blacksburg, Virginia

THE EFFECT OF INCONSISTENCIES IN A GRAPHICAL USER INTERFACE: OBJECTIVE AND SUBJECTIVE ASSESSMENTS

by
Richard H. Miller

Committee Chair: Robert J. Beaton
Human Factors Engineering Laboratory
Industrial & Systems Engineering

(ABSTRACT)

This research assessed the effect of inconsistencies in a graphic direct manipulation interface. A variety of measurement techniques were employed to determine how inconsistencies affected performance on an Apple Macintosh based computer application called “The Personal Organizer”. Three groups of 11 subjects each, all familiar with the Macintosh computer, were given a set of similar tasks on different versions of the application in a pretest (control version), treatment (control or one of two inconsistent versions), post-test (control version) experimental design.

Performance was measured using two objective measures, the time, and number of keyboard and mouse actions needed to complete forty-eight tasks. Subjects were administered twenty-nine semantic differential items for each version. Free response and yes-no questions on the consistency of twenty-two user interface components (for the treatment version only), direct comparison questions (for comparing the treatment and control version), and other free response questions were administered. Additionally, new guidelines and rules for creating GUI’s for the Macintosh are suggested from observational data collected during the experiment.

Results showed that time and subjective measures were not always in agreement on what was the most consistent interface. Users were sometimes not aware of the affect of inconsistences on their performance even though there

were able to detect them in the interface. Numerous rules and guidelines were proposed to help designers create consistent and more usable interfaces.

ACKNOWLEDGMENTS

The author would like to thank the committee members for their patience during this long endeavor. Chair Dr. Robert Beaton, Dr. James Wilson, and Prof. Paul Kemmerling, Jr. contributed to the success of this research.

Additionally, Dr. John Casali, Dr. Robert Williges, Dr. Jeff Woldstad, Dr. Walter W. Wierwille, Beverly Williges, and Dr. Terrell were very helpful as sounding boards for my statistical endeavors.

Fellow students, Dr. Sung Han, Dr. Kay C. Tan, Dr. Min Young Park, and Gerard Jorna also contributed to the project as well as to the authors state of mind. Additionally, I would like to thanks Dr. Bill Cushman, Wendy Kellogg, Dr. Peter Polson, Dr. Rex Hartson, and Andy Cohen for their advice. Martha Sprague, Kathy Menard, and especially Dr. Charles “Butch” Nunnally and David Sobotta were helpful in acquiring the necessary equipment to make the experiment flow smoothly.

Thanks to my mother, Betty Carr, and father, Robert Miller, for all their guidance and support during school and for bringing me up to have a strong desire to succeed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

Error!
is to design the user's model
Thomas P. Moran, 1981

The design of graphical computer interfaces involves many complex issues. The issue of consistency is a primary focus for creating usable interface designs. Consistency across applications allows users to transfer existing computer skills to new applications. This can dramatically reduce training time and errors, while increasing the user's opinion of the software's quality (Kellogg, 1987; Polson, 1988). In addition, the wide availability of various computer platforms fuels the need to have consistency across computer hardware.

Figure 1 shows ranges of consistency for a computer environment (adapted from Robert, 1988). An interface design that might be consistent in one program might not be consistent with other interfaces. There has been no research on how much emphasis should be given to each range of consistency when designing a user interface. It is unknown if the ranges reflect an order of relative importance (i.e., if it is more important to be consistent in range 1 than in range 2, etc…). It is not clear when specific consistency issues take priority in creating a consistent interface. Consistency can be created at one level at the expense of

creating inconsistencies at another level. The ability to determine where and how to make software consistent is an important issue to interface designers.

The present research looks at the effect of these components of consistency in an interface. The components occur within ranges 1 and 2 of Figure 1.

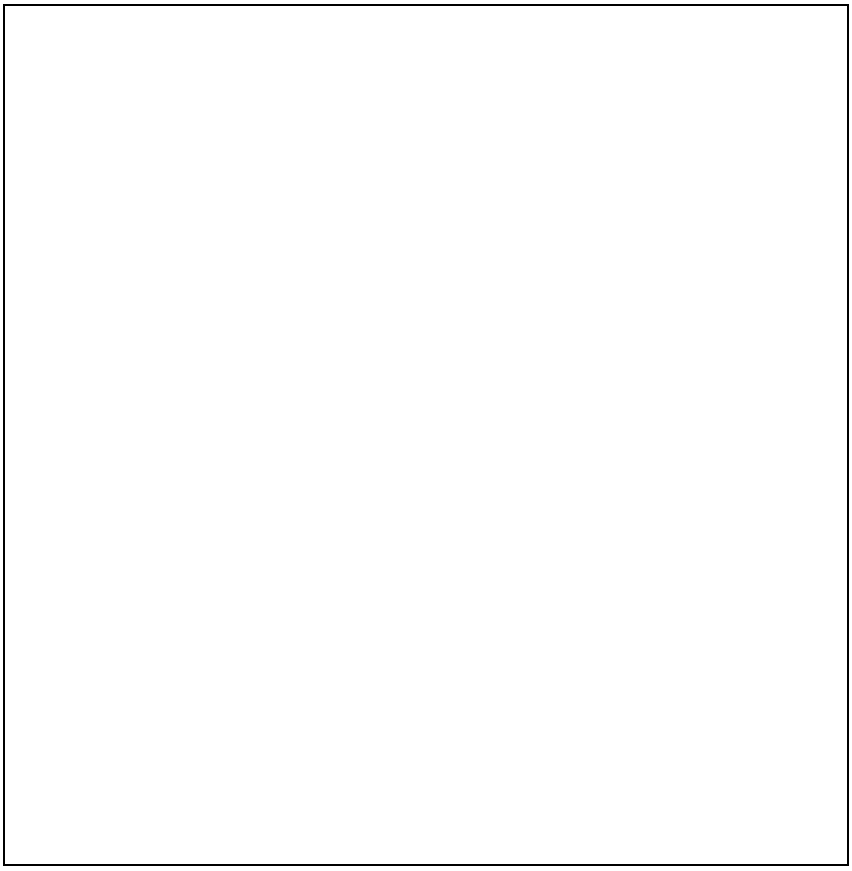


Figure 1. Different ranges of consistency in a computer environment (Adapted from Robert, 1988).

In general, consistency will be defined as; “system usage that meets the users’ expectations”. External tasks, in range 1, refer to the actions that allow for links to the system software or other applications found in range 2. Internal task

consistency would be used to define tasks specific to the application. Consistency with other systems, in range 3, has previously relied only on document conversion capabilities and not on functional or interface similarity. Today the trend is to make applications and documents transportable. Compatibility with the environment, in range 4, relies on consistency with external conventions. This is done by using metaphors in the software design. Other researchers have taken a different approach to defining the user interface.

Moran's Command Language Grammar (CLG) (1981) has provided a framework for researchers to stratify consistency components (Kellogg, 1987; Robert, 1988; Bouchard and Robert, 1989). Moran divides a user interface into three component parts, each with two levels. The overlap of the rings in Figure 2 reflects the interactions associated with the CLG. Polson stated that a strict decomposition of an interface into the six independent levels is probably futile; therefore, the overlapping areas. Polson (1989) pointed out that design decisions ripple through the CLG; and when changes occur at a one level, changes at other levels also occur.

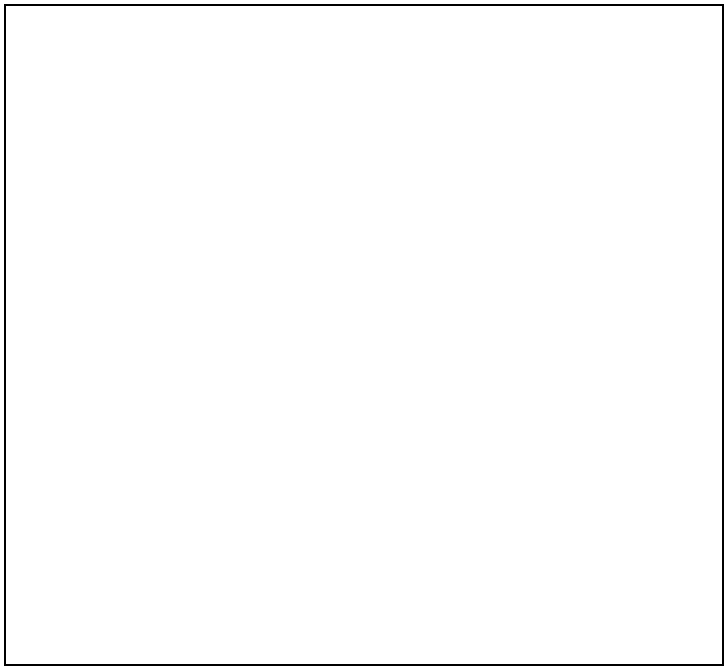


Figure 2. Moran's three components of a user interface.

Although it is difficult to separate issues into specific components and levels, it is possible to place some attributes into appropriate places in the taxonomy. The placement is based on where the primary effects occur in the design. It does not imply that the placement of a type of inconsistency into a specific level or component affects only that level. There is some overlap in the three components, but they can be distinctly defined.

The conceptual component consists of the task and semantic levels, both based on the abstract concepts of the system function. The task level organizes the user's needs into a structure that defines what events occur in the system. The semantic level defines the system's attributes and operation without defining the implementation.

The communication component consists of the syntactic and lexical level. The syntactic level recodes the semantic level concepts into actual commands, arguments, and state variables. The lexical level defines the actions and system rules needed to invoke the defined syntax.

The CLG shows the relationship between the conceptual and communication components. The physical component was not addressed in Moran's work, while Robert (1988) did use it to find inconsistencies in a user interface. The present research concentrates on the affect of all types of consistency in the CLG.

Clarification about how inconsistencies at these levels' impact objective measures and subjective opinion should help designers decide how to evaluate consistency and how to determine where to focus future research in interface design. Three versions of an application called "The Personal Organizer" were used in a pretest, treatment, posttest experimental design. Three groups of 11 subjects each, all familiar with the Apple Macintosh computer were used. Each group was given a set of similar tasks on a control version, followed by another set of tasks on the consistent (control version), or one of two inconsistent versions. The posttest followed with similar tasks using the control version for all three groups.

Subjective measures and time and keystroke data were used to identify differences in the affect of inconsistencies in three versions of a graphical direct manipulation interface. The versions varied in their communication, conceptual, and physical levels of consistency

1.1 Objective

The present research addressed the following questions concerning consistency in graphic user interface design.

* What types of inconsistencies do users recognize in an interface?
* How do time, keystroke and subjective measures differ with variations in the consistency of an interface?
* Do users recognize consistency as affecting their performance?
* Do users realize consistency affects their time to complete tasks?
* How do users describe inconsistent interfaces, what words can be used in conjunction with “consistency” to explain its affect?
* What rules, guidelines, or theories can be formed from observing experienced computer users with a new application?

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

Many technical areas contribute to the design of a computer application. General handbooks and guidebooks for user interface design (UID) can help create properly designed interfaces (Brown, 1988; Shneiderman, 1987; Smith and Mosier, 1984). Shneiderman's (1987) first golden rule of dialog design is to "strive for consistency". Shneiderman stated that "consistent sequences of actions should be required in similar situations, identical terminology should be used in prompts, menus, and help screens, and consistent commands should be employed throughout" (1987, p. 61). This rule is good, but applying it is difficult.

Some UI design guidelines were designed for specific computer environments. The Human Interface Guidelines: The Apple Desktop Interface (Apple, 1987) draws on some research for defining the implementation of the Macintosh environment. Digital Equipment Corporation published the XUI Style Guide (1988) to assist programmers in developing an X User Interface for DecWindows. IBM published Common User Access Panel Design and User Interaction (1987) for its System Application Architecture (SAA). Every computer company uses unique rules, regulations, and conventions for specifying their graphic user interfaces. Consistency is stressed in these works because it is an essential feature of an effective UID.

In a broad sense, rules are requirements; style guides show what a specific implementation looks like, while guidelines should be implemented if it "is reasonable and practical for you to apply the information to your application" (IBM, 1987, p. 3). Although each addresses a specific goal, they all have one

common focus, the consistency of the user interface. DEC's design manual is for programmers who want to create uniform and usable software in XUI applications to be consistent with other XUI applications (DEC, 1988). IBM's manual states, "the key to designing an effective interface is to develop in users the proper conceptual model of the interface as quickly as possible. The Common User Access does this through consistency" (p. 10, 1987). Finally, Apple asserts that "in most cases, consistency should be valued above idiosyncratic cleverness, " and that the "standard elements of the Apple Desktop Interface ensure consistency" (p. 6, 1987). These documents emphasize consistency with little or no empirical evidence to explain how much or what kinds of consistency impacts performance.

Designers can incorporate consistency in applications using a variety of methods. Lotus 1-2-3, the popular PC spreadsheet program, is a good example. Cohen and Schirmer's (1989) review of the successful Lotus 1-2-3 software package pointed out five major features that contributed to its dominance in the industry. One of these was "1-2-3's implementation of the <Escape> key [that gave] the user a consistent, non-destructive method of retracing his steps or correcting an error" (Cohen and Schirmer, 1989, p. 9). In reference to the CLG, the consistency in navigation represents two levels of the CLG. The conceptual consistency would be defined as a single step, available at any time, for retracing steps or correcting errors. The communication component would specify that the "Escape" key should always be used for implementing the semantic description.

In the DOS environment programs use different methods for creating a command syntax. Table 1 shows four common commands from IBM compatible programs; Cut, Copy, Paste, and Undo. Although most users probably use only one or two word processors, the examples are typical of programs written for DOS. Although some standards have emerged (F1 for Help, and the Escape key to up a level, or leave a mode), it was not until the Macintosh became popular that Microsoft Windows and DEC GEM emerged for IBM personal computers.

TABLE 1. Commands for Cut, Copy, Paste, and Undo in Various Applications

IBM PC Program	Cut	Copy	Paste	Undo
Word Perfect 5.0	Ctrl F4 then #1	Ctrl F4 then #2	Ctrl F4 then #3	Ctrl F4 then #4
MS Write	Shift+Del	Crtl+Ins	Shift+Ins	Alt+Bksp
Volkeswriter 3.0	Alt F8	none *	Alt F6	None
PFS Write	F4	F3	F6	Crtl F4

* Volkeswriter does not use a buffer for text, so only single pastes can be done. One cannot copy text and do multiple pastes without reselecting the text every time.

The Macintosh guidelines (Apple, 1987) suggest that Cut, Copy, Paste, Undo and other quick keys (keyboard short cuts for menu items) should be consistent across all applications in the Macintosh interface. All Macintosh programs that use these edit functions use the same set of keys (Command Z, X, C, V for Cut, Copy, Paste, and Undo, respectively). This means that despite the program one uses in the Macintosh environment, there usually will be a knowledge base that can be applied to the new application. “Macintosh users agree that learning new programs is almost intuitive. ‘At least I know how to get a program running…even if I don’t know exactly what it does’” (Voelcker, Wallich, Zorpette, 1986, p. 63). This is an example of applying consistency in Range 2 of Roberts levels of consistency.

Kieras and Bovair (1986) showed that learning time for a new system is reduced if the user is familiar with another system that has some rules in common. This also can be applied to applications found in the same system, as in the Macintosh. The above example illustrates the complexity of learning a new set of command keys for every application. This is less efficient that transferring

a single set of rules to new environments. Reisner (1981) working on similar research concluded that when “structural” consistency [the number of rules needed to define a system procedures] user performance was helped by consistent procedures. She observed that users made “expectation errors” in the treatment version, where they applied new, learned procedures to new objects. No formal analysis was performed.

Grudin, Ehrlich, and Shriner (1987) reviewed an internal Wang study of CAD system users that showed that consistency in the assignment of function keys to mouse buttons was not as critical as novice behavior indicated. No mention of “Criticalities” measures being either objective or subjective measures. Consistency, as defined by the designers, was then applied in situations where it would be beneficial. They avoided applying consistency across aspects of the interface where performance would not be affected by its absence. IT is unknown how performance was measured.

Carroll, Mack, and Kellogg (1988) discussed the use of interface metaphors in user interface design. The translation of non-computer tasks to computer-based tasks by using metaphors is an attempt to draw on previous knowledge and methodologies that might lead to a positive training effect. Since many problems can be transferred to, and solved with computers, the link between the external environment and the computer can be useful in helping users adapt. The ability to learn and use a product is usually based on previous experience. Whether the person has ever seen or even heard of the product, the user will use his or her experiences to form a knowledge base for new experiences. This knowledge base can stem from “typewriters, calculators, or video games, as well as from computer systems. A good user interface relies on these experiences” (IBM, 1987, p. 9). The use of metaphors creates links from this knowledge base to computer applications. Similarly, consistency within a specific application can be used to help a user map the knowledge of one function to untried tasks within the application. This mapping might involve a common metaphor, a specific interaction style, similar command syntax, or a common procedure.

Wolf (1989) pointed out that consistency is “composed of conflicting elements” (p. 89). In Figure 1 the ranges of consistency were defined, but by applying consistent design rules at any single level will lead to conflicts and inherent inconsistencies in other levels. Attempting to be consistent within an application, across applications, between platforms and among users is a wholly unworkable task. At most a designer can achieve consistency within the most important levels of the design. The direction and form of consistency is largely dependent on the goals the software designers set for the application. Many feel that the design concepts should be consistent with the end-user’s expectations (Blake, 1986; Grudin, 1989; Kellogg, 1987; Koritzinsky, 1989). This requires a designer to be familiar with the potential users of the system in addition to being able to have them assist in the design process. Goals might be a given level of subjective ratings or a minimum level of proficiency in completing a timed task. The present research addressed if both subjective and objective measures yield similar results.

It might be easy for a software engineer to design a set of menus to be consistently 5 centimeters wide and 10 items deep, but this may not be consistent with an end-user’s expectation for the design of the menu. Research addressing menu design considerations provides designers with guidelines to meet the expectations of users (Paap and Hofstand, 1986). The software engineer’s vision of consistency probably would not map to a user’s conceptual model. A novice computer user might be comfortable with drawing comparisons based on metaphors to restaurant menus (categorical, alphabetic or other organization methods), while experienced users would compare the menus to other applications and the workplace desktop (Apple, 1987).

Blake (1986) defined a consistent interface as having five characteristics:

* Predictable - Users anticipate what the system will do.
* Dependable - The system fulfills the user’s expectations.
* Habit-forming - The system encourages development of behavior patterns.
* Transferable - Habits developed in one context apply in new situations.
* Natural - The interface is consistent with the user’s understanding.

These attributes can be part of both the communication, conceptual, and physical components of an interface. All could be used as synonyms for

consistency. The present research addressed what types of words are used in a similar fashion as “consistent” or “inconsistent”. This can help designers understand how to apply a comment that a feature was “natural” or “predictable”. Other researchers have attempted to define consistency.

In 1988, a group of experts in human computer interaction (HCI) gathered to discuss consistency but could not create a good definition (Grudin, 1989). Researchers have created numerous methods for explaining and quantifying HCI. Some researchers have found generic methodologies for comparing interface designs (Bachman, 1989; Roberts and Moran, 1982). Tullis (1983) developed a metric for determining layout complexity, while Card, Moran and Newell (1983) developed the well known Keystroke Level Model (KLM). Reisner (1981) used formal grammatical descriptions to predict ease of use and Robert and Wang (1989) continue to build statistical metrics for evaluating the importance of inconsistencies. Others have tried to quantify and classify consistency (Barnard, Hammond, Morton, Long, and Clarke, 1981; Berry, 1988; Bouchard and Robert, 1989; Robert 1988). Although these efforts are laudable, true taxonomies for defining consistency are still only works in progress. The present research does not propose a taxonomy, but an analysis of what features make an interface “consistent”.

Even a simple definition of consistency is difficult to apply. A dictionary defines consistency as a “logical connection…agreement with what has already been done or expressed; conformity with previous practice” (Mish, 1983, p. 303). Often, a designer may be trying to match an interactive style to a user’s cognitive representation of the system’s workings. The user’s representation might not include any previous computer practice and would be based only on experiences from non-computer activities (like reading a restaurant menu or using a typewriter). This type of agreement can play an important role in the learning process. Bouchard and Robert defined consistency as “the quality of the system which respects the users’ expectations” (p. 2, 1989).

Koritzinsky (1989) concluded that it was more important to be consistent with the user’s perspective of the interface than to constrain the shape of a new icon. The author was referring to the addition of a clock icon as more appropriate as a

circle, since a user thinks of most clocks as circular than square. A square would make it consistent with the other system icons, while a circle would be consistent with the shape of watches and wall clocks. No experimental evidence was cited to indicate whether the shape of the object would make any subjective or objective difference in user performance. Making the clock consistent with "square" digital clocks was not addressed.

Ham (1987) stated that "no matter how much trouble it is, take pains to be consistent in every way possible: punctuation, significance of colors, mode of input, location of messages…" (1987, p. 114). Grudin (1989) probably would not agree with Ham. Grudin pointed out many instances where consistency is not the most important issue to be resolved. Most agree that it is better to strive for consistency. Ham, a programmer, clearly addresses the need for an interface to be consistent with the users model. Unfortunately, Ham's statement is problematic, since it might not be realistic to be consistent in every possible way. Something consistent in one context probably will be inconsistent from a different point of view. In addition, there are costs to being consistent: user testing; the system timeline; and designer effort might have to be increased. A possible solution is to allocate resources toward creating consistency in aspects of the interface where performance would be affected by its absence, than creating consistency in every aspect of the interface. Grudin (1989) suggests that one should center effort on understanding the tasks of the worker, than on just being consistent. Grudin pointed out that consistency is not the goal, but only a method for achieving usability in a design. An appropriate design should allow for consistency within the context of the workers' tasks.

Hammer, Kunin, and Schoichet (1983) concluded that a consistent interface is one "in which different things are handled in similar ways with similar syntax" (p. 129). The word "things" referred to the object in a verb-object syntax, where actions were performed on system objects. Their paper was developed to assist designers in using a conceptual framework for evaluating or constructing a UI. They cite four criteria for designing a UI. They say an interface should be natural, easy to learn, easy to use, and most importantly consistent. They go into very little depth on the four criteria, and do not have any empirical data to support

their research. They stated that “consistency is the cornerstone of a system that is easy to work with, easy to learn, and easy to remember” (p. 130). According to them, consistency is an important aspect of the conceptual framework of a user interface design.

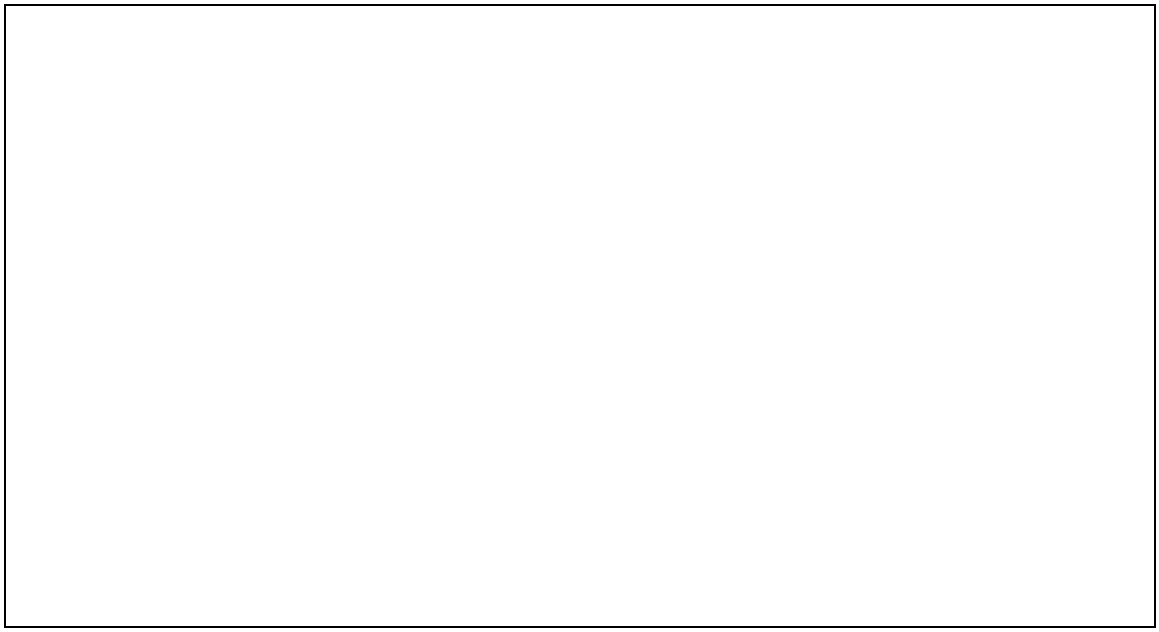
Briggs (1989) investigated the use of a type of diagram, the command wheel, and proposed that it can “provide a suitable meta-language with which to illustrate the consistent properties of a command set” (p. 160). The study looked at the consistency of WordPerfect and Word, both commercially successful word processors. The command wheel is used to map out the word processors’ commands for cursor control and deletion tasks. Briggs does not define the command wheel, but explains it by example. The command wheel method was good for that study, but is difficult to adapt for commands beyond the cursor control paradigm. Briggs questioned if consistency was dependent on a person’s awareness of the underlying rule structure, or if a user would benefit from consistency even if he or she were not aware of the inconsistency in the GUI. This is an important point that needs to be addressed.

Whiteside, Jones, Levy, and Wixon (1985) looked at user performance with command, menu, and iconic interfaces. They tested a system as a whole because it “represents a design team’s best overall solution to optimizing many variables, [and] we claim that a test of the system as a whole is more fair and valid than a (perhaps impossible) decomposition into component variables” (1985, p. 185). They suggest that, due to the inordinate complexity of a user interface, attempts at decomposition are probably futile. The present research takes a similar holistic approach, as an entire design’s faults are measured, not one specific attribute.

Robert (1988) presented a method for detecting inconsistencies in user interfaces. The first step entailed defining the system, a computer-based bibliographic information retrieval system using a production system with IF (user action) THEN (system response) type descriptions. Robert found that this method, by itself, was not effective for detecting inconsistencies, and expanded on the six description levels from the Command Line Grammar (CLG) model

proposed by Moran(1981) to evaluate the interface. Moran's model has three main components: the Conceptual, Communication, and Physical, as discussed in the INTRODUCTION. Each main component has two underlying levels. Robert views the levels as "six different angles for examining…precise issues where consistency comparisons can be made" (1988, p.360). Table 2 shows the three components, six levels, and some interface components analyzed in the bibliographic database system.

TABLE 2. A Framework for Detecting Inconsistency (Robert, 1988)



The attributes in Table 3 were generated by comparing differences in interface components found in the same level. For example, Robert defined command names that were verbs, nouns, adjectives, expressions, etc., as the "type of identification" into the lexical level. When command names were abbreviated differently, the "type of abbreviation" was created as a lexical attribute.

TABLE 3. Sample Attributes of a Bibliographic System at Three Levels of Moran's CLG (Adapted from Robert, 1988)

Syntactic	number of arguments position of arguments default option (to skip steps) shortcuts (to go faster) command nesting (grouping commands under one name) recursion (to use same command w/o selecting it again) scope of recursion (over all or some arguments) time at which commands are shown, vanished, dimmed, or highlighted
Lexical	representation (iconic, alphabetic, numeric) identification (noun, verb, expression) language (English, French) number of words, word length and abbreviations type of delimiters and type styles blinking and blinking speeds abbreviations, capitalization, font size and style
Spatial	position on a support (visual display, keyboard, mouse) menu item position (grouping vs. separation) alignment, indentation

Robert did not show a way to determine the number or type of attributes. He says, "to a large extent, the number of attributes that is considered may determine the power of the method for detecting inconsistencies" (1988, p. 360). No previously created list of possible attributes was referenced, and identification of the attributes was dependent on the awareness of the evaluators.

Once these attributes were determined, differences within the same category and across each attribute at specific levels were compared. Any differences

between the components were labeled as possible inconsistencies. Finally, differences were classified as inconsistencies "as long as the classification made sense in the context of the task. Indeed, the task was used as the ultimate guide because it is responsible, to a large extent, for the structure and operation of a system" (1988, p. 360). Although not systematically repeatable, this heuristic method is one of the few examples in the literature that tests inconsistencies in UID. Over fifty inconsistencies were detected with this classification scheme. In an experiment with ten subjects who looked for inconsistencies in the system for 75 minutes, a total of 18 different inconsistencies were found. All had been previously uncovered by the classification scheme.

On a subjective level, all but one subject reported that the system was easy to learn and use, with a few negative feelings. So, while there were many inconsistencies, as defined by Robert's method, it was not reflected in the subjective evaluations of the users. The potential decrement to performance from the inconsistencies is unknown; user performance (i.e., time to complete a task, number of errors) was not measured. It is unclear if the inconsistencies not found by the subjects were not salient enough to be noticed, masked by other features of the interface, or not within the scope of the users' objectives. This research leaves many questions unanswered. Of the eighteen inconsistencies found by the users, eight were classified in the communication component, eight in the conceptual component, and the remaining two in the physical component. So, it is unclear which component contributed more in the subjective ratings. Robert did conclude that the method was robust for detecting inconsistencies.

Kellogg's (1987) experiment addressed consistency in the user interface and its effect on user performance using Moran's (1981) CLG framework. In Robert's (1988) work the levels were crossed with specific components of the interface (Table 2) while Kellogg completes the matrix by dividing the interface components into two distinct groups, internal and external consistency at the conceptual level. Kellogg looked at conceptual and not procedural consistency. Procedural consistency is generally found in design guidelines (e.g., the assignment of function keys or the placement of a menu item). Conceptual consistency was defined in terms of the "internal coherence of a system's

structure and the nature of the mapping from user-level goals to system procedures" (Kellogg, 1987, p. 389).

In Kellogg (1987) three groups of users were each assigned two sets of tasks, each group having a different version of the interface with analogous tasks. Versions were consistent, inconsistent, and consistent plus salient. The versions all performed the same functions, but varied in the semantic procedure needed to access the features. In the first trial a significant difference was found between the inconsistent and the two consistent interfaces, but in the second trial (when training became a factor) the effects were not significant. Subjects also recalled more correct procedures from the consistent interfaces and recalled fewer inappropriate procedures. An important result from this experiment was that the subjects' rated the consistent interfaces more positively on fifteen of seventeen bipolar adjective scales taken from Coleman, Williges and Wixon (1985). The only drawback was the lack of power for finding significant differences ($p < .05$) in the time to task completion in the second and third trial. The lack of power can primarily be attributed to the small sample size. Kellogg summarized that users of the consistent system performed tasks more quickly, recalled system procedures almost perfectly, and correctly inferred untried procedures. Users in the consistent group also had greater confidence in their abilities to use the system.

Kellogg's conclusions were similar to Payne and Green's (1986) findings that consistency in system design allowed users to generalize rules and actions from known procedures to novel tasks. In addition, Payne and Green concluded that consistency can impact the performance of users without their knowledge of its effect. In an experiment on syntactic consistency, Payne and Green (1986) found that a high-level consistent syntax for three types of tasks was the easiest to learn of three syntax structures tested. A second design incorporating two rule structures used in the same tasks was harder to learn than a design with a distinct command structure for each category. The ability to use a consistent set of syntax in a variety of tasks proved to be the most effective method for inter-application consistency.

Polson (1988) reviewed experiments that addressed the acquisition, transfer, and retention of user skills. Kieras and Polson (1985) proposed the Cognitive Complexity Theory (CCT), which based subjects' learning and performance of a computer based system on the number of novel and known rules needed to define the tasks of the system. CCT uses production rules based on a GOMS model (Card, Moran, and Newell, 1983) to quantify the content and structure of knowledge which underlies computer-based skills.

Polson, Muncher and Engelbeck (1986) accounted for 88% of the variance in the mean training times of subjects to complete tasks. This was based on the formula training time equals n*t + c, where n is the number of new rules to be learned to master a task, t is the training time per rule, and c is a constant specific to the training procedure. Individual times were predicted with a model based on the subject's mean time to completion, number of new rules, and the number of newly generalized rules. This model accounted for 70% of the variance in individual training times. The training time for a newly generalized rule was 3.5 seconds, which implies that there was a time cost associated with the transference process. When multiplied by thousands of tasks this could be a considerable amount of time.

The research reviewed generally used only one measure; objective or subjective. The present research addressed how subjective and objective measures differed among versions of a single application. It was hypothesized that the type of inconsistencies noticed by subjects will vary among the versions of the interface. The ability of subjects to notice inconsistencies and the type of adjectives used to describe those inconsistencies will also be documented.

Some research stressed the importance of minimizing the type and number of rules that subjects need to learn to be useful. This research tried to determine rules, and guidelines, that would be considered consistent by the users of the system. It was predicted that many of the rules used by designers were arbitrary or nebulous and not well enough defined to be used in creating a direct manipulation interface.

In conclusion, consistency and it's effect on the creation of usable interface designs is poorly defined and not well understood. The general viewpoint is that consistency is good, but it is not known how to determine where and in what

context something should be consistent. Also, the use of information from users' in a software design process is warranted, but knowing when to ask for their input, and what questions should be addressed, has not been adequately researched. The present research used a form of the CLG and user involvement to determining how consistency affected performance in a graphical user interfaces.

CHAPTER 3 METHOD

3.1 Overview

Computer users (faculty, staff, graduate and undergraduate students) from the Virginia Polytechnic Institute & State University area, took a screening test to gauge their ability to understand Macintosh commands and procedures. Subjects who passed the screening test and a vision test participated in a controlled laboratory experiment.

Three versions of an application were developed for the experiment. The application, called “The Personal Organizer”, consisted of a datebook, notepad, calendar, computer file facsimile transmitter, address book, area-code finder and telephone dialer.

The first version was the control version for the experiment. It was refined by a combination of human factors engineers (HFE) and subjects using an iterative design process. Once this version was refined during five months of development, two more versions were created based on this first design, both varying in the conceptual, physical and communication components of consistency. Both inconsistent versions incorporated a multitude of design variations.

Eleven subjects were assigned to each of three sets of tasks on the control (consistent) version of the application. A set consisted of two blocks of eight tasks. Subjects then completed two more blocks of tasks on either the consistent or inconsistent versions of the interface depending on the group that they were assigned. After completion of the experimental treatment all users were again

given the control version of the application and asked to perform two more blocks of eight tasks.

The time to complete the tasks and the number of steps needed to complete each task was recorded in all trials. After completing each set, a questionnaire was administered. Rest breaks were allowed after each block. Participants were compensated for the 2 ½ hour session with a combination of shareware diskettes and money.

3.2 Subjects

Thirty-three subjects, 29 males and 4 females, participated in this study. Each subject spoke English as his or her native language, and read and signed an informed consent form (See Appendix A). Sixteen undergraduate students, twelve graduate level students, three faculty members, and two staff members participated. Subjects averaged about 4 years of Macintosh experience. The age of the participants ranged from 18 to 39, with a mean age of 25 years.

One hundred and twelve subjects "experienced" Macintosh users were administered a screening test to determine their level of proficiency on the Macintosh computer. A priori assumptions were not made about the subjects' level of expertise, although experience with the Macintosh and HyperCard was required to pass the test. The screening test was designed by HFE's and pilot subjects during the iterative design of the control version of the application. Passing was defined as the top scoring thirty-three subjects that were administered the test. The screening test is outlined in Appendix B. These questions were chosen to represent an experienced Macintosh user's knowledge.

Experienced subjects were selected under the assumption that novice users would not be familiar enough with the Macintosh to recognize inconsistencies. Without a solid working knowledge of a variety of Macintosh software and extensive use with the platform, subjects would be unable to distinguish what would be considered inconsistent with other Macintosh applications. The assumption that experienced users notice more inconsistencies in an interface was not tested in this experiment. Additionally, novice users were used during

the development of the application. Due to the experimental design the inclusion of novices would have either caused participation time to increase, or the number and complexity of the tasks to be decreased. Either scenario was considered unsatisfactory. The difficulty in scheduling subjects' for multiple sessions, and potential maturation effects ruled out multiple sessions. Decreasing the number of tasks would have limited a subject's exposure to the interface giving them even less information about how to make judgements about the consistency of the interface.

Each subject participated for approximately 2½ hours and received compensation of just software (fifteen 800K disks of shareware), or six disks and six dollars. The offer of software as compensation was thought to be an attractive alternate to standard monetary compensation. Many subjects expressed great interest in the software.

3.3 Materials and Apparatus

The applications were created using SuperCard 1.5, which is similar to HyperCard. Hardware consisted of a Macintosh IIfx with a Radius high resolution color monitor (1152 x 882 pixel addressability), an 8-bit color board, 8 megabytes of RAM, and an Apple mouse for user input. The Macintosh was operated in the MultiFinder mode running system software version 6.0.5.

The application, called "The Personal Organizer", simulates a datebook, notepad, calendar, facsimile transmitter, address book, area-code finder and telephone dialer. It was a single functional program and not a prototype.

There were three versions of the application. Functionally, all application versions performed the same operations. The consistent version had "no" inconsistencies. It was refined and iterated by HFE's and pilot subjects, and defined as consistent for this experiment. This was done by designing the control version to follow existing guidelines as close as possible (Apple, 1987; Apple, 1989). Twelve pilot subjects, each with five to fifteen hours of experience with the application, were used to refine the consistent version. In addition, human factors engineers were used to help complete the designs. It was assumed that any inconsistencies that still existed in the control version, after it

was refined, would exist in all subsequent inconsistent versions and not bias a specific version of the interface. Defining the first version as consistent was a relative measure. The control version was defined as consistent for this experiment, although some “inconsistencies” were known to exist in the control version of the interface.

Each inconsistent version embodied a unique set of inconsistencies. Both varied in their communication, physical and conceptual consistency as defined by Moran (1981). Some of these variations were similar in both inconsistent versions of the interface, in other cases it varied between all three versions. Sometimes a consistent “state” could only have one inconsistent state. For example, the Find function remembered the previously found word in the consistent version. In both inconsistent versions the Find function did not remember the previously listed text. Both inconsistent versions varied similarly in this “inconsistency”. Appendix G outlines the known differences between the three versions of the interface.

The Questionnaire for User Interface Satisfaction (QUIS), developed by Chin, Diehl, and Norman (1988), was used, in a modified form, to rate the interfaces after each of the three sets. Bowers (1990) cited others that reported users of the QUIS were not reading every question when the scale ranged from left to right (negative implication to positive). The questions were bipolar semantic differentials on a 1 to 7 rating scale. Bowers modified the structure of the scale without changing the questions and still obtained high Cronbach Alpha values (over .80’s). Bowers’ concluded that users’ were more thoughtful in their answers. So, the order of the questions and the position of the positive and negative anchors to the scale were randomly assigned for each questionnaire. All subjects’ received the same three questionnaires in the same order (See Appendix C for a sample of the questionnaire). The questions were used to determine if consistency affected the opinions of the subjects. The questions were also used to generate a list of adjectives that users found to be similar to consistency.

At the completion of the three sets of tasks and questionnaires, an additional set of subjective measures consisting of bipolar scales, and interview style questions were

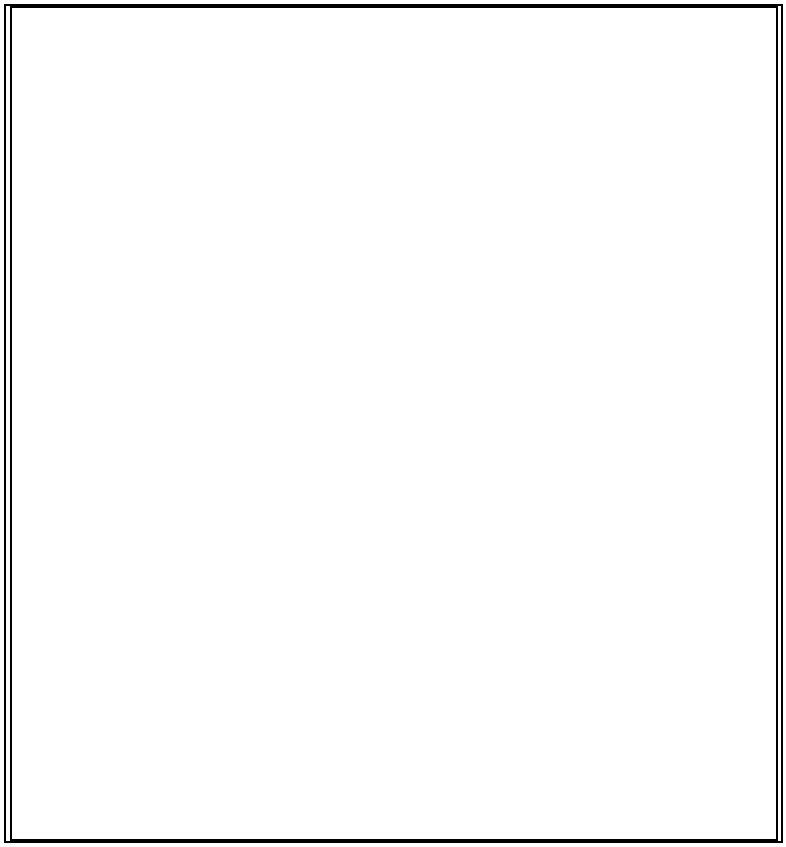
administered (Appendix D). Bipolar scales were used to collect preference data for the application and the interface. An unstructured set of twenty-two questions at the end of the experiment were used to elicit information about specific attributes of the treatment version that may have been missed by the bipolar scales. The answers were used to assess the type and number of inconsistencies identified by the subjects. Additional questions were asked to obtain the subject's age, type of computer use, and computer experience.

3.4 Experimental Design

Subjects were randomly assigned to a group (Figure 3). The first set of tasks (a set consists of two blocks of eight tasks each and a post-set questionnaire) lasted approximately 60 minutes. The next two sets lasted less than 30 minutes, with rest periods after every set. The blocks of trials used in the tests were similar and almost identical across sets. For example, one task in each block is for placing a name, address, and phone number in the Address Rolodex, but each occurrence of that task incorporated a different name, address, and phone number. Since the blocks were "almost" identical and not exactly identical they were presented in a random order for each subject, additionally tasks within Blocks were also randomly ordered for each Block and each subject.

Although the sets of tasks (two blocks) were identical the two blocks of tasks within each set were not necessarily identical. The types of tasks found in each block are defined in Table 4. The reason for choosing a variety of tasks was twofold. The subjects could experience more nuances of the interface and become more familiar with the consistency or inconsistency of the interface when exposed to a variety of tasks. Second by having more tasks repeated less frequently, than less tasks repeated more frequently, it would be less likely that the subjects' would become bored. Subjects were also able to learn about more aspects of the interface without being able to anticipate the next task. Of course, it was very difficult to "match" tasks within and across sets. Since blocks were matched, subjects can only have one block of each type in each set. So if a subject received an odd numbered block first then they could only receive block 2, 4 or 6 next. This gives each subject the same sixteen tasks for each version

of the interface. The tasks were matched (A to B, C to D, E to F, etc…) by two measures; the number of steps required to complete the tasks, and the time required to complete the tasks. Typing was always minimal, but when it was required the time to complete the matched tasks, and the number of keystrokes was kept constant. Each block requires a minimum of 58 steps (actions recorded by the computer), and about 15,600 ticks (about 4:20 minutes).[1] This type of performance could be achieved by a moderately skilled computer operator (30 wpm.) who knew how to perform each task.



[1] ticks are $\frac{1}{60}$ of a second

Figure 3. The experimental design.

TABLE 4. Description of Tasks Found in the Blocks of Tasks.

Tasks for Blocks 1, 3 and 5	Tasks for Blocks 2, 4 and 6

A- Move a meeting and place a conference in the Weekly Planner.	B- Place a meeting in the weekly planner twice (two weeks apart).
C- Find a phone number and other information and enter it into two places in the Weekly Planner, without going to the Address Rolodex.	**D-** Find the correct area code for a specific city and place it in front of both phone number found on two Address Rolodex cards that show that city as the address.
E- Add or change two settings in the Phone Setup function.	**F-** Sort the Address Rolodex by last name.
G- Delete two specified pages of the To Do function and add some comments in a specific size and font style to a specific page.	**H-** Add two pages to the To Do function at a specific place in the function and add some wording in a specific size and font style to a specific page.
I- Add a set of cities to a specific region in the Area Code function.	**J-** Add an address card to the Rolodex with a person's name, company, address, and office phone number.
K- Delete the Address cards for three individuals.	**L-** Find a persons telephone number, without going to the Address Rolodex, and dial the phone number with a specific phone setting from a specified page in the Weekly Planner or To Do function.
M- Correct the name of a company as it appears in the weekly planner, with the correct information found in the Address Rolodex.	**N-** Find the correct area code for a phone number, specified in the Weekly Planner, by comparing it to what was in the Area Code function.
O- Get help for a specific icon in a specific function.	**P-** Get help for a specific icon in a specific function.

Tasks were matched across rows for time and steps, even if the length of the instruction is short the time and steps to completion are equal.

The performance-based dependent measures, time and the number of steps to complete each task, were recorded. These were computed from the sum of the individual task completion time and step data. Subjects were not allowed to go to the next task until they had completed the previous task. Less than 1% of the tasks were not completed correctly (less than 16 out of 1584). No correction factor was employed to account for the small amount of uncompleted tasks, since subjects' would usually just miss one small part of the task (i.e., not making a word Bold etc…). Steps were defined into one of 3 groups; Completed with quick keys (command keys), Completed with menu selections (using the mouse), or Completed by clicking on icons (with the mouse). Steps were not classified any further due to the difficulty in assigning a step to a type of error (i.e., how does one define steps taken to complete the tasks "the long way", or when the subject completes all the steps up to a point correctly and then starts to explore other alternatives that also could lead to the correct result). These measures will determine if subjects were affected by inconsistencies as well as if the subjects interaction with the interface changed as a function of the consistencies of the interface.

An ANOVA was performed on a 3x3x2 mixed factor design. A control group was used to account for learning effects associated with using similar software applications. Groups (3 levels) and Subjects within Groups were the between subject variables. Set (3 levels) and Block (2 levels) were the within subject variables. Set and Block were both completely crossed with the between subject variables. A block consisted of a set of eight tasks as outlined previously in Table 4. The version that a Group received in Set 2 defines the Group's name. Accordingly Group 1, 2 and 3 received versions 1 (Control), 2 (inconsistent version A), and 3 (inconsistent version B), respectively.

Blocks in Set 1 were training sessions to allow groups to become familiar with the interface. This helped decrease the differences between groups before the onset of the treatment version. It also gave the subjects a basis for making judgements on inconsistencies found in the treatment version. Set 2 exposed Group 2 and 3 to the inconsistent versions of the interface. Any changes in performance between groups could then be associated with the onset of the inconsistent versions. Set 3 was used to see if future performance would be affected by transfer of training from the inconsistent versions in Set 2.

A short subjective evaluation (Appendix D) was performed after each trial. The subject rated the interface on a set of twenty-nine bipolar semantic differential scales. The data was analyzed using an ANOVA procedure on a subset of the questionnaire. Since the twenty-nine scales were given after each set, and were identical, this allowed for a comparison of the changes in subjective ratings due to the version presented in Set 2. It also was useful for determining a set of adjectives that reflected subjects interpretation of consistency.

After the last trial the subjects filled out a questionnaire about his/her background and computer experience (Appendix E). The subject was allowed to analyze the version that the subject used during the second set to assist in completing the questionnaire in Appendix E. Free response questions addressed the consistency of the interface. Subjects were given twenty-two topics to define as either consistent or inconsistent. Subjects were given space to provide an explanation for their answer. This allowed for an analysis of subjects ability to identify specific inconsistencies in the interface.

During the entire session the subjects were observed by the experimenter. Numerous theories and rules were formulated while observing the subjects and pretest subjects. These are proposed to help develop future interface styles, rules, and guidelines. These should help define specific attributes of consistency.

3.5 Experimental Procedure

3.5.1 Pre-Experimental

Subjects were informed that the experiment was for evaluating software and the user interface, but consistency was never specifically mentioned. They were then asked to sign an informed consent form (see Appendix A). Upon completion of the informed consent form and after passing the Macintosh “vision” test and keyboard setup procedure outlined by the instructions in Appendix F, the actual experiment began. The intent of the vision/keyboard test was to make sure the user was comfortable with reading small italic type (a worst case scenario), and in distinguishing the colors red, green, blue, yellow, and black (used in the experimental conditions). The mouse, keyboard, screen, and chair

were adjusted to suit the subject, although the mouse and keyboard response rates were not changed. The screen highlight color was a light blue, and the keyboard repeat rate was set at the third fastest setting. Subjects could position the mouse on the left or right side of the keyboard. All subjects read the passage, and could distinguish the colors.

3.5.2 Experimental

Subjects were given eight tasks, in a random order, from one of the six sets of tasks. The experimenter intervened only when absolutely necessary and requested by the subject, or when the subject incorrectly assumed that they had completed the exercise correctly. If a subject was unable to continue the experimenter said either, “Use the Help system if you are having problems”, or “Please reread the question and make sure you are doing everything the question has asked for”. If that did not help the subject (and it did most of the time), then the experimenter would single out the part of the question that the subject did not understand by reading the question to the subject and emphasizing the misunderstood part. For example, if the subject did not “**bold**” a piece of the text that the task requested to be bold, and the first attempt at assistance did not work, then the experimenter could read the question back to the subject with the emphasis on the word bold; “…Put “Movies to Rent” in 18 pt bold type at the top of the left visible page…”. Only in a few (two) instances did the experimenter have to add additional comments for the subject to complete the task (out of a total of 1584 tasks).

After the first two blocks of tasks were completed, the first post-test questionnaire was administered. Since the tasks in the next four blocks were similar to those already completed, subjects required almost no interjections from the experimenter. On average, each subject “forgot” how to complete one or two of the previous tasks and needed the occasional, “You have not completed the task as specified” followed by “Please reread the question to verify that you have completed everything the question has asked for”.

Once the subject completed the six sets of blocks (with appropriate rest breaks between blocks) the final questionnaire was administered. The subject was instructed to “play with” the second version of the interface when trying to answer

the questions. The subjects were not allowed to use the application when completing the prior questionnaires, but were encouraged to explore this version while filling out this questionnaire. They were made aware that no specific tasks were to be performed and no video or computer based data was being collected during the questionnaire period.

3.5.3 Data Collection

The application automatically logged the time and input from the user. A VHS camera recorded the screen actions and any comments made during the session as a backup, it was not analyzed. The software recorded user inputs in increments of $\frac{1}{60}$ of a second, a tick. Although all keystrokes were not recorded, keystrokes that changed the users mode of operation, or point of reference were recorded. For example, if the user typed in a field the act of entering the field was a counted as a step, but the actual text typed while in the field was not recorded. From observing subjects, one could see that typing made up a very small proportion of the total time required to complete the task. If the subject selected a piece of the text while still in the field and selected bold, it was recorded. Figure 4 {Parts A & B} outline a sample of the actual information that was recorded by the software during the completion of a task.

Instruction Presented:

Go to pages 5 and 6 of the To Do function and add two more pages. Put "Movies to Rent" in 18 pt bold type at the top of the left visible page. Put "Die Hard, Mask, Batman, Star Wars, Animal House" under it in 18 pt. plain type.

Steps Taken to Complete Instruction:

Step #	Time Started	How and Where Step was Completed
0	20069	•••Done! with EndVersion Instruction 008 Card 49•••
1	20680	w/Icon bkgnd button "To Do" of bkgnd "Weekly" of window "Date"
2	20820	w/Icon bkgnd button "Next" of bkgnd "To Do" of window "Date"
3	20936	w/Icon bkgnd button "Next" of bkgnd "To Do" of window "Date"
4	21456	w/Icon item "New Pages" menu "Edit"
5	21615	w/icon bkgnd field "todoL" of bkgnd "To Do" of window "Date"
6	21765	w/Menu Size item "18" menu "Size"
7	21905	w/Menu Style item "Bold" menu "Style"
8	22206	w/Key Style item "Plain" menu "Style"
9	23118	w/Icon bkgnd button "Return_Overview" of bkgnd "To Do" of window "Date"
10	23154	Done! withEventH3 Instruction 09 Card 44[aaa]

Explanation of Steps Taken to Complete Instruction - Subject:

1) clicked the "To Do" button in the Main window,
2-3) clicked the next button twice to go from page 1/2 to 5/6,
4) clicked the New Pages icon (if he had used the menu it would have said "w/menu"),
5) clicked in the field and typed the text into the field (steps requiring typing text were observed and not recorded by the program),
6) selected all the text and selected size "18" from the menu using the mouse,
7) selected the Bold menu choice using the mouse,
8) selected the necessary text and used the keyboard equivalent for "Plain" text (note the w/Key prefix),
9-10) returned to the main window and clicked the DONE! button.

Figure 4-A. A sample of the steps required to complete Task H of Block 6.

Statistics Recorded From this Task:
Time to complete task (23154-20069) = 3085 (about 51 seconds)
Number of steps to complete task =10
Number of steps using Icons/Quick Keys/ Menu Choices= 6/1/2 [does not include clicking the DONE! button]

Statistics Known About this Task:
Minimum number of steps needed to complete task=8
Minimum amount of time needed to complete task= 1600 (about 27 seconds)

Fastest and Least Steps Method for Completing this Task;
[Task starts with the completion of the last task]
1) Go to the To Do function by clicking on the icon or by using the function key,
2) “Find” page 6 {with the Icon or Quick Key, this moves the subjects to page 6},
3) New page {with icon or menu or quick key},
4) Enter field {by clicking mouse in area of the field},
5) Type all text- select the text- and select 18 point from menu {this is considered one step since once one enters a field the next action must be outside of the current focus of the field},
6) Select the text to be bold and choose the Bold menu choice { the rest of the text should already be plain, but some subjects verify this by selecting the text to be plain and choosing plain from the menu, this is considered an extra step},
7) Finally the subject clicks the return to main screen button,
8) Then clicks the DONE! button. [This completes the task and starts the next task.]

Figure 4-B. A sample of the steps required to complete Task H of Block 6.

CHAPTER 4 RESULTS

The results of this experiment were divided into three main categories; objective performance (time and steps), subjective discrete response, and subjective free response measures. The results based on observing the subjects are presented in the Discussion section. An analysis of variance (ANOVA) was used to test for time differences due to Set, Block, and Group conditions. Simple effect tests were performed on significant two way interactions. Post-hoc comparisons using a Newman-Keuls (N-K) test were calculated on those effects found to be significant in the main and simple effect tests. The probability of Type-1 error less than 0.05 was established as significant for the ANOVA. A Spearman correlation between the Time and Step data was calculated. Further analysis on the Step data was performed using a Kruskal-Wallis analysis.

Subjective measures were analyzed using ANOVA and nonparametric statistical tests. The Mann-Whitney (MW) test was used for comparisons between groups, and the Wilcoxon signed rank test was used for within group comparisons. Spearman rank-order correlation coefficients were used as a measure of association for the sets of bipolar semantic differential items.

4.1 Time and Step Data Analysis

Two dependent measures, time and steps were measured. A Spearman correlation between time and steps was .807 ($p<.0001$). Figure 5 shows a scattergram of the time/error rates recorded for each subject (33 subjects x 6 blocks = 198 observations). The step data was the combination of the number of keyboard clicks, clicks on icons with the mouse, and menu selections with the mouse. A partial correlation analysis using the

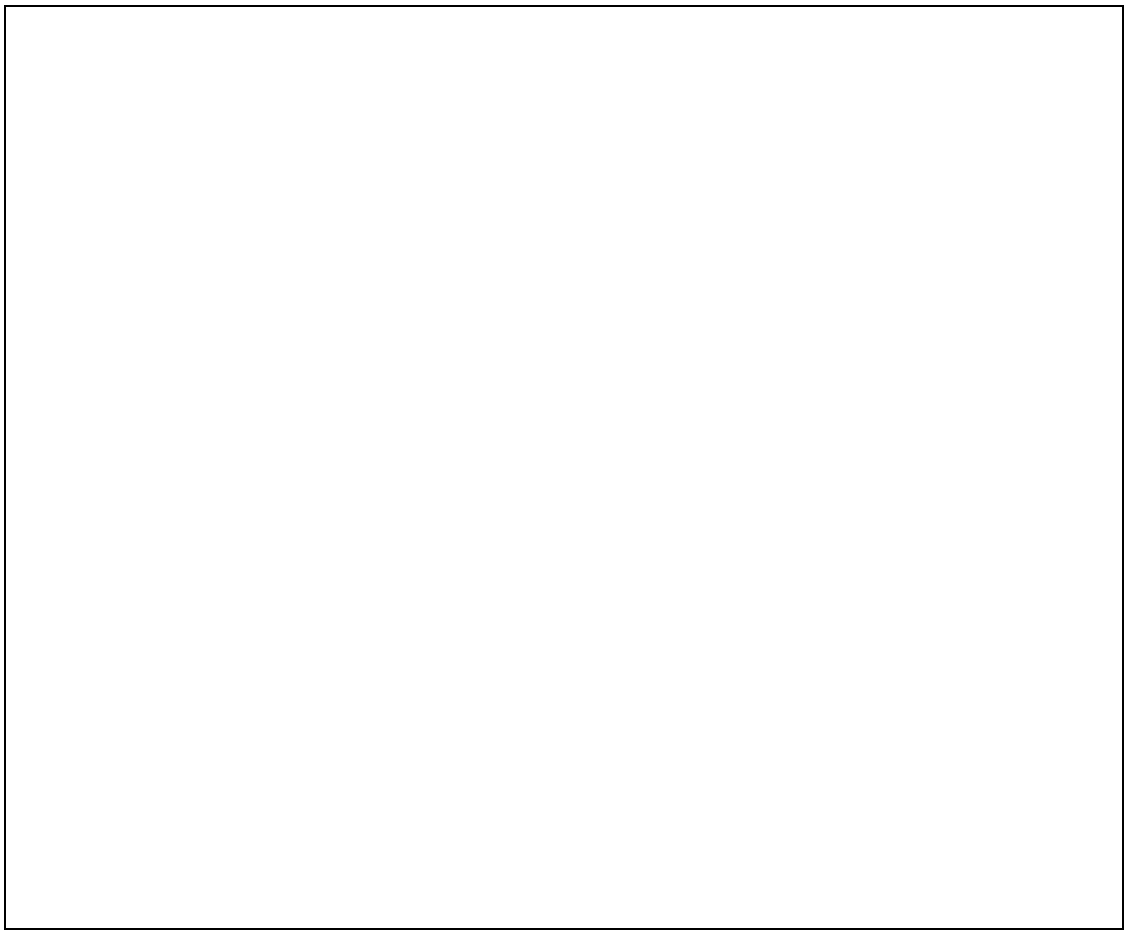


Figure 5. Relationship between time and step data (n=198).

Pearson Correlation Coefficients revealed icon usage alone accounted for 95.5 percent of the variance in the Step data. Keyboard and menu data had much smaller correlations with the total number of steps (23.1 and 35.1 percent, respectively). This information was useful for revealing changes in the subjects interaction with the application due to inconsistencies. The data reflected that subjects did not change their interaction style. For example, the mean number of menu selections was 5.36, 5.64, and 6.05 for the three groups during their use of the second version. This difference is not significant in a Kruskal-Wallis analysis ($p=.6927$). The changes in the Step data can be attributed to the decrease in the number of steps completed by using the mouse to click on icons. Due to the high

correlation of the step and time data, and the high partial correlation for icon usage to total Steps, only the time data was analyzed.

The ANOVA used to reveal differences in completion time between groups revealed five significant results (Table 5). The main effect Group was not significant, but that was expected given the experimental design. A N-K test was not needed on the Block main effect, it had only two levels, thus the difference between Block 1 and 2 was statistically significant, but not practically significant. The significant main effect Set was treated to a N-K analysis (Table 6). The differences are of little use since they are confounded by the three Set 2 treatment conditions. A Simple effect test for Block at each level of set showed performance significantly improved in Set 1, but not in Set 2 or 3 (Table 7). Blocks in Set 1 were considered "training" to help equalize possible between group differences. Set 2 was used to test differences due to the effect of inconsistencies. Data from Set 3 could have indicated if any halo effects, or transfer of training influenced the time performance measure.

The three-way Block by Set by Group effect also was not significant. This would have been useful to analyzed if Groups were experiencing very large differences due to the version of the application. In this experiment, the differences only extended to the first presentation of the inconsistent version of the application in the second Set. The differences in the interfaces were not large enough to keep the subjects from overcoming the effect of inconsistencies. Furthermore, the subjects showed no residual effect from being exposed to the inconsistent version when performing tasks in the control version in Set 3 (Table 8). That table does reveal that groups did show differences in the Set 1 condition (Table 9). Normally the differences in Set 1 would preclude making direct comparisons of the differences found in Set 2, but further analysis revealed that comparisons were possible. Figure 6 clearly shows that the differences between groups in Set 1 can be attributed to the variance in the first Block of that Set. Groups in Set 1 differed by about 328 seconds in Bloc 1 (Set 1), but only 32 seconds in Block 2 (Set 1). Subjects who could be considered "outliers" in the first Block where not necessarily the same subjects who had poor performance in Block 2. Since the instructions required every task to be completed correctly subjects might have spent an inordinate amount of time on one or two tasks

accounting for their slow times in Block 1. The primary statistics of interest is not that all groups improved in their performance (Table 11) for each successive Set (Tables 12, 13, and 14, for Groups 1, 2, and 3, respectively), but if their was any differences in performance between Groups in Set 2. Figure 7 illustrates the differences found in Set 2. The N-K comparison showed that the Groups who used an inconsistent version of the interface did exhibit significantly slower performance that the control Group (Table 10). This was useful in establishing that the versions were different on an objective metric, so further subjective assessments could be compared to this result.

Analysis of the Block by Group interaction is included for completeness (Table 15, 16 and 17). The Block by Set interaction is also presented (Table 18, 19, and 20). The statistically significance of these results reveal no practical significance. The differences between Sets is explained in the previous analyses, and this only again shows that the Blocks within the Sets gave the subjects enough opportunities to improve their performance on the application.

TABLE 5. Analysis of Variance Summary Table for Performance (Time) Data

Source of Variance	df	MS	F	p
Between Subjects				
Group (G)	2	8.342E+08	1.93	0.1627
Subjects/Group(Su/V)	30	4.322E+08		
Within Subjects				
Set (S)	2	1.492E+10	151.12	0.0001
Set*Group(S*G)	4	2.597E+08	2.63	0.0431
Set*Subj/Group(S*Su/G)	60	9.876E+07		
Block (B)	1	5.722E+09	75.95	0.0001
Block*Group(B*V)	2	4.272E+08	5.67	0.0081
Block*Subj/Group(B*Su/G)	30	7.535E+07		
Block*Set	2	2.723E+09	34.25	0.0001
Block*Set*Group(B*S*G)	4	1.565E+08	1.97	0.1108
Block*Set*Subj/Group(B*S*Su/G)	60	7.951E+07		
Total	197			

TABLE 6. Newman-Keuls Rating Comparisons for Main Effect Set

SET	Mean Time (in ticks)			
1	68314	A		
2	47045		B	
3	39265			C

Note: Means with the same letter are not significantly different, p<.05.

TABLE 7. Analysis of the Simple Mean Effects of Factor Block

Source	df	MS	F	P
Block at Set 1	1	1.078E+10	135.58	<.01
Block at Set 2	1	276070913	3.47	NS
Block at Set 3	1	111904297	1.41	NS
BLK*S*Su/G	60	7.951E+07		

TABLE 8. Analysis of the Simple Mean Effects of Factor Group

Source	df	MS	F Test	P
Group at Set 1	2	7.093E+08	7.18	<.005
Group at Set 2	2	5.903E+08	5.98	<.005
Group at Set 3	2	54028604	.55	NS
S* Su/G	60	9.876E+07		

TABLE 9. Newman-Keuls Rating Comparisons for Group at Set 1

Group	Mean Time (in ticks)		
3	63944	A	
1	66264	A	
2	74731		B

Note: Means with the same letter are not significantly different, p<.05.

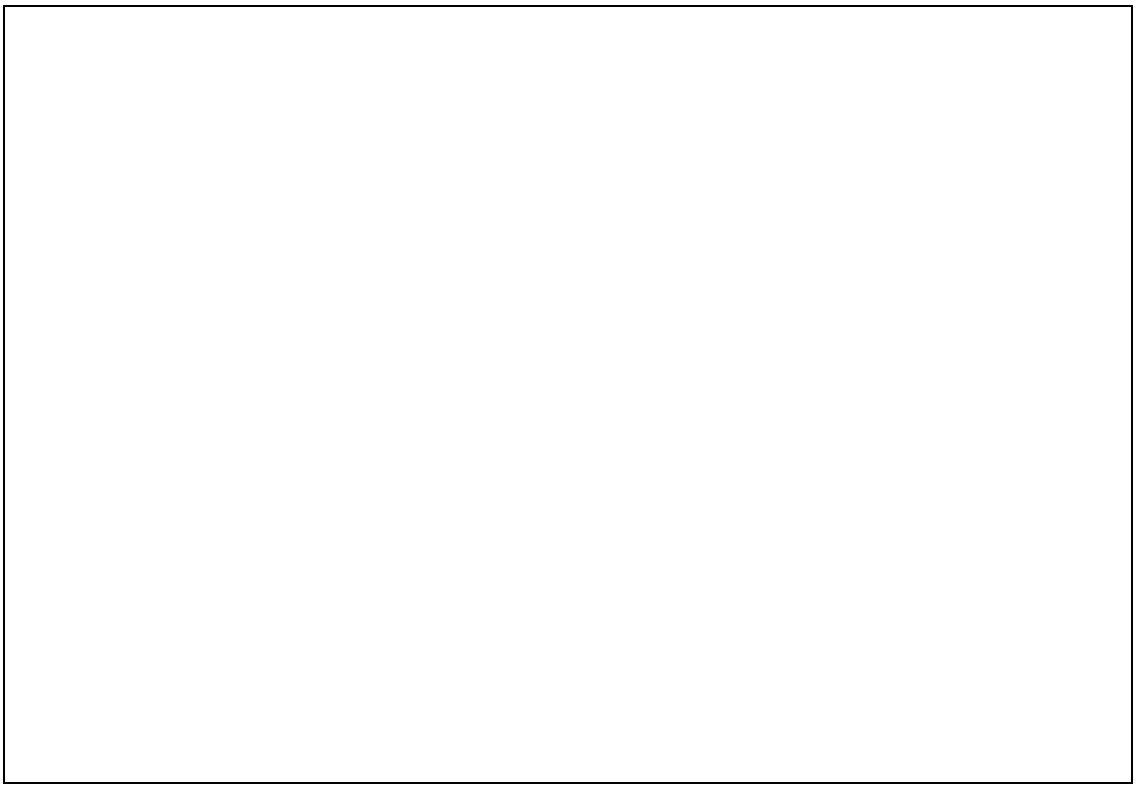


Figure 6. Time to complete Blocks for each Group.

TABLE 10. Newman-Keuls Rating Comparisons for Group at Set 2

Group	Mean Time (in ticks)		
1	41475	A	
3	47940		B
2	51718		B

Note: Means with the same letter are not significantly different, $p<.05$.

TABLE 11. Analysis of the Simple Mean Effects of Factor Set

Source	df	MS	F Test	P
Set at Group 1	2	4.819E+09	48.80	<.001
Set at Group 2	2	6.718E+09	68.02	<.001
Set at Group 3	2	3.907E+09	39.56	<.001
S* Su/G	60	9.876E+07		

TABLE 12. Newman-Keuls Rating Comparisons for Group 1 Between Sets

Set	Mean Time (in ticks)			
3	39861	A		
2	41475		B	
1	66264			C

Note: Means with the same letter are not significantly different, p<.05.

TABLE 13. Newman-Keuls Rating Comparisons for Group 2 Between Sets

Set	Mean Time (in ticks)			
3	40446	A		
2	51718		B	
1	74731			C

Note: Means with the same letter are not significantly different, $p<.05$.

TABLE 14. Newman-Keuls Rating Comparisons for Group 3 Between Sets

Set	Mean Time (in ticks)		
3	37486	A	
2	47940		B

1	63944	C

Note: Means with the same letter are not significantly different, $p<.05$.

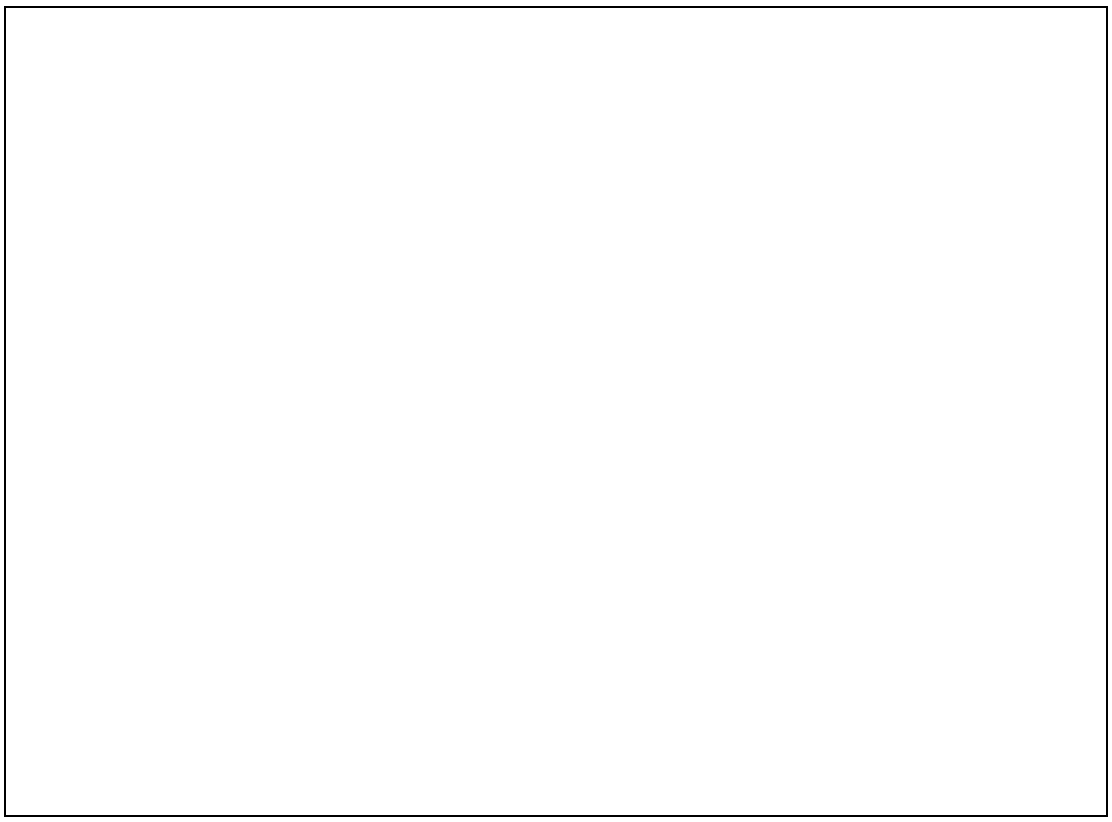


Figure 7. Mean Time to complete Sets for each Group.

TABLE 15. Analysis of the Simple Mean Effects of Factor Block at Levels of Group

Source	df	MS	F	P
Block at Group 1	1	1.064E+9	14.12	<.001
Block at Group 2	1	4.559E+9	60.50	<.001
Block at Group 3	1	953442833	12.65	<.005
BLK*Su/G	30	7.535E+07		

TABLE 16. Analysis of the Simple Mean Effects of Factor Group at Levels of Block

Source	df	MS	F	P
Group at Block 1	2	1.0223E+9	16.23	<.001
Group at Block 2	2	38411853	0.51	NS
BLK*Su/G	30	7.535E+07		

TABLE 17. Newman-Keuls Rating Comparisons for Block 1 Between Groups

Group	Mean Time (in ticks)		
1	53216	A	
3	53591	A	
2	63943		B

Note: Means with the same letter are not significantly different, p<.05.

TABLE 18. Analysis of the Simple Mean Effects of Factor Set at Levels of Block

Source	df	MS	F	P
Set at Block 1	2	1.507E+10	189.54	<.001
Set at Block 2	2	2.581E+9	32.46	<.001
BLK*S*Su/G	60	7.951E+07		

TABLE 19. Newman-Keuls Rating Comparisons for Block 1 Between Sets

Set	Mean Time (in ticks)			
3	40566	A		
2	49090		B	
1	81094			C

Note: Means with the same letter are not significantly different, p<.05.

TABLE 20. Newman-Keuls Rating Comparisons for Block 2 Between Sets

Set	Mean Time (in ticks)			
3	37962	A		
2	44999		B	
1	55533			C

Note: Means with the same letter are not significantly different, p<.05.

4.2 Subjective Questionnaire (Discrete Response)

4.2.1 Preference Index

The twenty-nine questions used in the first post-test questionnaire were used to determine the bipolar adjective pairs that best represented the type of information sought in this experiment. The Consistent-Inconsistent (C-I) item was correlated with the other twenty-eight items. Correlated items showed that they were rated similarly, and indeed if the correlations were perfect we could look at only a single item to base our conclusions. Since, the anchors used to describe the scales were different, we can consider them as testing a different, but similar, set of preferences.

As a measure of subjects internal consistency a Cronbach Alpha test was performed. It was high for all three sets of questionnaires (.922, .951, .922 for Sets 1, 2, and 3, respectively). This suggests that groups rated the interface consistently. All items (after rotation)[2] had positive correlations with the C-I item. Only those items that had a correlation with the consistency scale of greater than .35 ($p < .05$) were used in formulating a Preference Index (PI). Of the twenty-nine items fourteen were chosen to create the PI. The linear sum of the scores on the fourteen scales could range from 14 to 98. The data used to generate the PI was obtained from the questionnaires administered after Set 1. Table 21 lists the twenty-nine bipolar adjective pairs used in creating the PI. The items are listed in the same order as in the first questionnaire with each pair ranging from “good trait” to “bad trait”. Note all adjective pairs were positively correlated with the C-I item. Each subject received three questionnaires, thus three PI scores.

[2] So all items went from 7 (the good/consistent end of the scale) to 1 (the bad/inconsistent end of the scale).

TABLE 21. Spearman Correlation Coefficient (r_S) for Bipolar Adjective Pairs with the Consistent-Inconsistent Item in the PI

Q #	Bipolar Items		r_s	p
1)	Pleasing	Irritating	.445	.0118 *
2)	Friendly	Unfriendly	.273	.123
3)	Complete	Incomplete	.141	.4262
4)	Cooperative	Uncooperative	.463	.0088 *
5)	Dependable	Undependable	.618	.0005 *
6)	Simple	Complicated	.577	.0011 *
7)	*Consistent*	*Inconsistent*	---	----
8)	Natural	Unnatural	.324	.0666
9)	Intelligent	Unintelligent	.298	.0914
10)	Interpretable	Uninterpretable	.388	.0283 *
11)	Fast	Slow	.101	.567
12)	Adaptive	Unadaptive	.001	.995
13)	Useful	Useless	.301	.0885
14)	Redundant	Concise	.226	.2003
15)	Uncluttered	Cluttered	.166	.3479
16)	Safe	Unsafe	.009	.9616
17)	Maintainable	Unmaintainable	.353	.0459 *
18)	Easy to learn	Difficult to learn	.474	.0074 *
19)	Not frustrated	Frustrated	.463	.0088 *
20)	Productive	Unproductive	.482	.0064 *
21)	worth buying	not worth buying	.344	.0517
22)	Relaxed	Tense	.34	.0547
23)	Optimistic	Pessimistic	.525	.003 *
24)	Competent	Incompetent	.429	.0153 *
25)	Pleased	Disgusted	.458	.0096 *
26)	Engaged	Bored	.296	.0945
27)	Happy	Unhappy	.054	.7617
28)	Intelligent	Stupid	.203	.2517
29)	Pleasant	Annoyed	.463	.0088 *

* Sig. at .05 level, and $r_S > .35$

An ANOVA was performed on the PI data (Table 22), and a N-K test was performed on the significant main effect Set (Table 23). This reveals that Groups did recognize the interfaces presented in Set 2 as different. The Set by Group interaction demonstrates that there was no difference between Sets for subjects in Group 1 (the control group). It also shows that Groups 2 and 3 recognized that the interfaces were different that what was presented in Sets 1 and 3. A simple effects test was performed on the significant interaction effect for Group at the three levels for Set (Table 24). The significant effect from that test was further analyzed with a N-K test (Table 25). A simple effects test was performed on the significant interaction effect for Set at the three levels of Group (Table 26). The two significant levels, in the simple effect test, were treated to a N-K test (Table 27 and 28).

TABLE 22. Analysis of Variance Summary Table for Preference Index (PI)

Source of Variance	df	MS	F	p
Between Subjects				
Group (G)	2	448.859	1.28	0.2929
Subjects/Group (Su/G)	30	350.808		
Within Subjects				
Set (S)	2	525.525	10.50	0.0011
Set*Group (S*G)	4	132.0859	2.64	0.0425
Set*Subj/Group (S*Su/G)	60	50.066		
Total	98			

TABLE 23. Newman-Keuls Rating Comparisons of PI for Set

Set	Mean PI		
3	79.273	A	
1	78.697	A	
2	72.091		B

Note: Means with the same letter are not significantly different, p<.05.

TABLE 24. Simple Effects Test for Group: Set Conditions for PI

Source	df	MS	F Test	P
Group at Set 1	2	82.303	1.649	NS
Group at Set 2	2	563.546	11.256	<.0001
Group at Set 3	2	67.182	1.342	NS
S*Su/G (S*Su/G)	60	50.066		

TABLE 25. Newman-Keuls Rating Comparisons of PI for Group at Set 2

Group	Mean PI			
1	78.909	A		
2	72.727		B	
3	64.636			C

Note: Means with the same letter are not significantly different, p<.05.

TABLE 26. Simple Effects Test for Set: Group Conditions for PI

Source	df	MS	F Test	P
Set at Group 1	2	6.576	0.131	NS
Set at Group 2	2	271.485	5.423	<0.01
Set at Group 3	2	511.636	10.219	<0.001
Set*Subj/Version (S*Su/V)	60	50.066		

TABLE 27. Newman-Keuls Rating Comparisons of PI for Set at Group 2

Set	Mean PI		
3	82.000	A	
1	80.455	A	
2	72.727		B

Note: Means with the same letter are not significantly different, $p<.05$.

TABLE 28. Newman-Keuls Rating Comparisons of PI for Set at Group 3

Set	Mean PI		
3	77.182	A	
1	75.545	A	
2	64.636		B

Note: Means with the same letter are not significantly different, $p<.05$.

4.2.2 Comparison of Preference Index Within Groups

The data is presented in Figure 8 to show the magnitude of the drops in PI for the groups who had the inconsistent versions of the interface in Set 2. Subjects in the Version 1 Group were very consistent in their rating of the three identical versions of the interface and showed no significant difference in their ratings. The drop in the PI between Set 1 to 2 and Set 2 to 3 for Groups 2 and 3 were significant (Table 27 and 28). Figure 8 is useful in recognizing the difference in PI for Sets in each Group. Groups rated that interfaces identical in the first and third sets. No difference in the first set was expected since the groups received the identical version. The nonsignificant difference between groups in Set 3 implies that Group 2 and 3 did not experience any negative transfer of training from having an inconsistent version for the interface in the second Set.

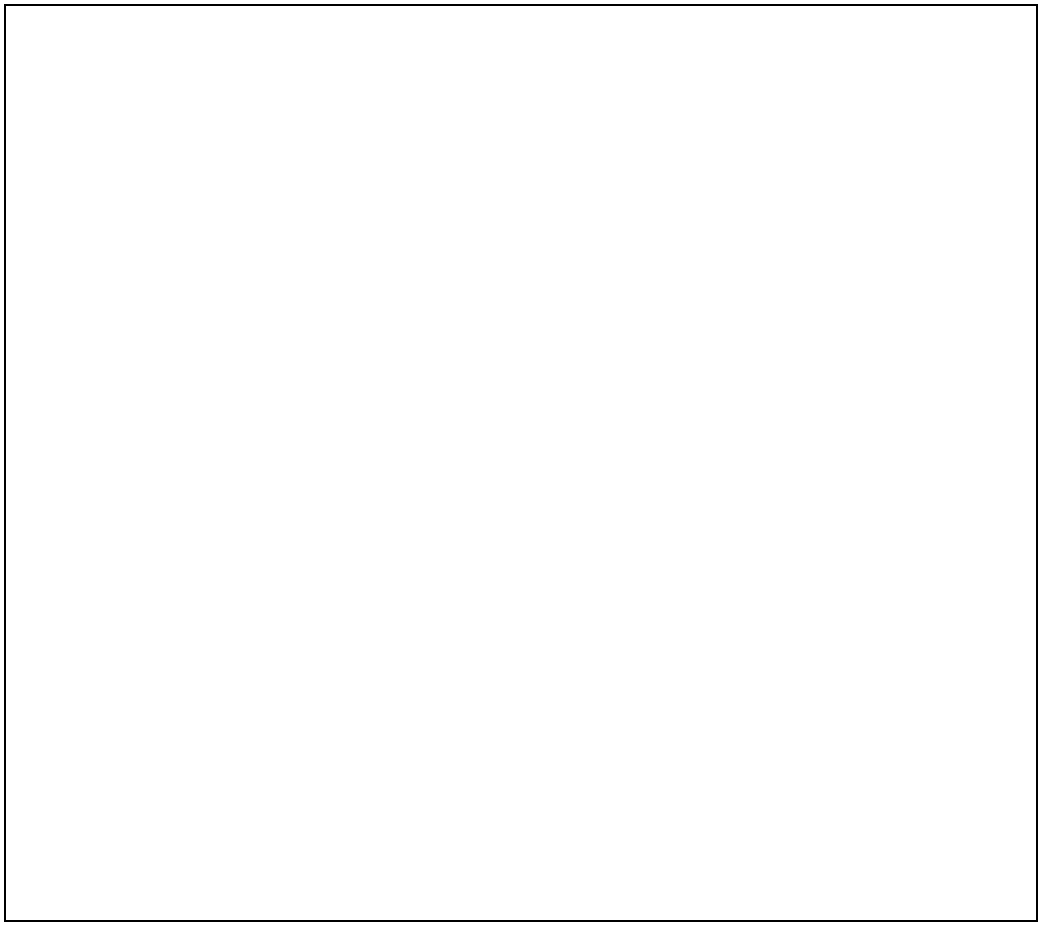


Figure 8. Variations in the Preference Index as a function of Set and Group.

4.2.3 Kruskal-Wallis One Way ANOVA for Consistent-Inconsistent Ratings

After completing the final post-test questionnaire the subjects were given a final questionnaire (Appendix D). The final twenty-two questions concerned the consistency of of the application presented in Set 2. Subjects circled “consistent” or “inconsistent” for each topic. Space was provided for subjects to explain their answers. A score of 1 was assigned to a rating of consistent, and a score of 2 for inconsistent. When the subject did not rate an item, or circled both consistent and inconsistent, a score of 1.5 was assigned (neither inconsistent or consistent). This method for dealing with missing or ambiguous answers decreases the variability of the sums making the analysis more conservative. A Mann-Whitney U test performed on the sum of each subjects ratings revealed significant differences between subjects in Group 1 and 2 ($p= .0299$) and between Groups 1 and 3 ($p=.0008$) (Table 29). This shows that subjects did recognize the inconsistent interfaces as being more inconsistent in specific areas.

It was also interesting to note the number of questions that were assigned a 1.5 rating. Subjects in Group 1 rated twenty seven answers as neither consistent or inconsistent out of the 242 (11 subjects * 22 topics for each subject) questions. Subjects in Group 2 (9 ambiguous answers) and Group 3 (8 answers) had less of a problem identifying the topics as being either consistent or inconsistent.

TABLE 29. Mann-Whitney U Comparisons Between Groups for the Consistent-Inconsistent Ratings of Interface Topics

Comparison	Σ Rank of score for Groups in comparison	*p* Value
Groups 1 vs. Groups 2	93.5 vs. 159.5	.0299 *
Groups 2 vs. Groups 3	105.5 vs. 147.5	.1639
Groups 1 vs. Groups 3	75.5 vs. 177.5	.0008 *

* significant at $p < .05$ *p* values are corrected for ties

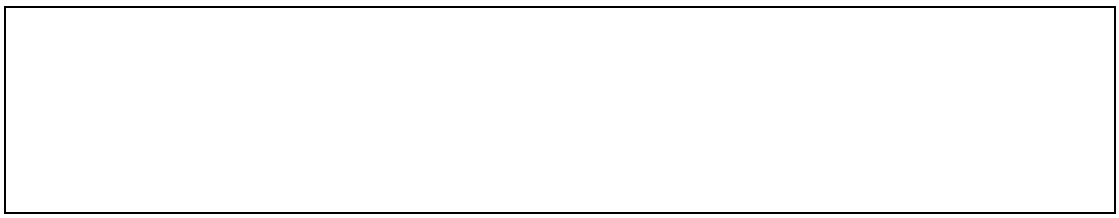
4.2.4 Correlation Between Subjective Estimates and Actual Completion Times

In the final questionnaire subjects were asked, “How many minutes do you think it took to complete each of the six sessions?” This data was helpful in determining if consistency affected the subject’s perception of time. Pearson product moment correlations were computed to compare the post-test estimates of time to complete the six blocks and the actual completion times.

The clock in the consistent version was always visible in the top right corner of each function. Inconsistent version A contained the clock in the Calendar and Weekly Planner, while version B had the clock in the Navigator in addition to the Calendar and Weekly Planner. Note, in Table 30, the low correlation for the subjects who were only administered the consistent version three consecutive times, and similarly high correlations for the two groups that were administered inconsistent versions of the application during the treatment condition. Actual

times were in ticks ($\frac{1}{60}$ of a second) and predicted times were in minutes. Both groups that had inconsistent versions showed significant correlations.[3]

TABLE 30. A Comparison of Correlation Coefficients for the Real and Predicted Time to Set Completion for the Three Groups



4.2.5 An Analysis of the Direct Comparisons Questions

Three questions asked the subjects, “Overall, how would your compare the three versions of the program?. They were asked to respond on a 1 to 7 scale anchored with “Equal” and “Different” for the first & second, second & third, and first & third versions of the interface. These questions were asked during the post experiment questionnaire (Appendix D). The treatment version was displayed on the screen while these questions were answered. Answers from each combination of groups were compared using a Mann-Whitney U test. To compensate for any inflation of the α level, due to the multiple comparisons, $p<.017$ was treated as significant for all comparisons.[4]

The interpretation of Table 31 is complicated. For example, $p= .0006$ for the top left cell indicated that subjects who received the control version for both the first and second sets rated the versions differently than the subjects in Group 2,

---

[3] Since N is large (66), significance was tested using the t distribution using the formula $t = r_S \sqrt{\frac{N - 2}{1 - r_S^2}}$

[4] Solving for familywise error FW = [1 - (1-α)^3] with α = .05 yields FW α = .017.

who received the control and treatment versions in the first and second set. The results in the third column showed that all groups found the versions in the first and third sets to be identical (all groups' received the control version in both the first and third sets). The most important finding, in Row 2, indicated differences between the two groups 2 and 3, were not significantly different for any comparisons. Both Groups 2 and 3 found differences in both comparisons, the Set 1 & 2 comparison, and the Set 2 & 3 comparison. This reflects that the subject did notice that they were using different interfaces, but that the two inconsistent versions were not judged at different from each other.

TABLE 31. Mann-Whitney U comparisons between versions for the direct comparison questions

Comparison	Question 1 (Compares Set 1 & 2)	Question 2 (Set 2 & 3)	Question 3 (Set 1 & 3)
Group 1 & Group 2	.0006 *	.0002 *	.6838
Group 2 & Group 3	.4611	.5902	.8593
Group 1 & Group 3	.001 *	.0001 *	.4273

* significant at $p < .017$ p values are corrected for ties

4.2.6 Comparision Questions Between Versions in Sets Two and Three

Four questions were asked in the post experiment questionnaire (Appendix D) for directly comparing the version in Set 2 and 3. The questions are presented in

Figure 9. The lower the mean value the "better" they would rate the consistent over the treatment version. A mean score of 4 would imply the second and third version were equal.

An analysis similar to that performed on the twenty-nine bipolar adjective pairs was performed for these questions. Since all scales were significantly correlated with the More Consistent-Less Consistent scale (Table 32), the linear sum of the five measures was taken to create a five item Direct Comparision Index (DCI). Table 33 shows that comparisons between Group 1 and 2 and Group 2 and 3 were significantly different. Unlike the comparisons in Table 31 these questions did find Groups 2 and 3 as rating the interfaces used in Set 2 as significantly different. Although not statistically significant, the sum of the DCI scores for Group 3 is *higher* than the DCI for subjects in Group 1 (the control group) and implies the inconsistent version B was as consistent as the control version! This is juxtaposed to the within group analysis for the PI that did reveal differences (Figure 8).

How would you compare the third version of the program to the second?

Better			Second Version			Worse
1	2	3	4	5	6	7

Confusing			Second Version			Understandable
1	2	3	4	5	6	7

Easier			Second Version			Harder
1	2	3	4	5	6	7

More Consistent			Second Version			Less Consistent
1	2	3	4	5	6	7

Simpler			Second Version			Complicated

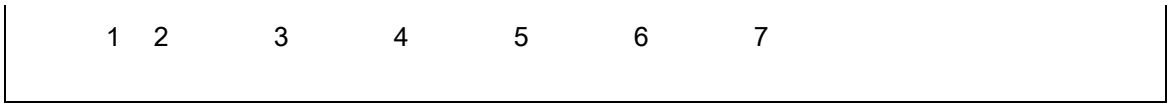


Figure 9. Direct comparison questions.

TABLE 32. Spearman Correlation Coefficient (r_S) for Each Bipolar Adjective Pair with the More Consistent- Less Inconsistent Scale for the Direct Comparison Index

Q #	Bipolar Scale **		r_S	p
1)	Better	Worse	.6921	.0001*
2)	Understandable	Confusing	.6955	.0001*
3)	Easier	Harder	.7079	.0001*
4)	More Consistent	Less Consistent	-----	------
5)	Simpler	Complicated	.6714	.0001*

* Sig. at .05 level, and $r_S > .35$

TABLE 33. Mann-Whitney U Comparisons Between Groups for the Direct Comparison Index

Comparison	Σ Rank of PI-5 score for Groups in comparison	*p* Value
Groups 1 vs. Groups 2	167 vs 86	.007 *
Groups 2 vs. Groups 3	87.5 vs 165.5	.0101 *
Groups 1 vs. Groups 3	108.5 vs 144.5	.2199

* significant at $p < .05$ *p* values are corrected for ties

4.3 Subjective Questionnaire (Free Response)

4.3.1 Comments Made in 22 Free Response Consistent/Inconsistent Questions

When asked to mention consistencies or inconsistencies in twenty two areas concerning the interface, the subjects in Group 3 were more descriptive and used a significantly higher number of words to describe the interface that subjects in Group 1 (Table 34). Subjects used a total of 1906, 2224, and 2796 words to describe the treatment versions (Groups 1, 2, and 3), respectively. This indicates that subjects found it easier to express what was wrong about the interface that what was right. It could also imply that users were trying to "guess" the true purpose of the experiment (to identify inconsistencies) and thus did not feel the need to explain when they defined a topic as consistent.

TABLE 34. Mann-Whitney U Comparisons Between Groups for the Number of Words Used to Describe Inconsistencies on Twenty-two Free Response Questions

Comparison	Σ Rank for Groups in comparison	*p* Value
Groups 1 vs. Groups 2	108 vs 145	.2244
Groups 2 vs. Groups 3	111.5 vs 141.5	.3245
Groups 1 vs. Groups 3	96.5 vs 156.5	.0488 *

* significant at $p < .05$ *p* values are corrected for ties

The comments were scrutinized to extract “usable” comments. Comments like; “I did not like the Find function” {too general}, “It was ugly” {not relevant}, or “I did like one way simple for it” {gibberish} were not classified. The remaining comments relating to the consistency of the interface were classified according to a scheme developed from Moran’s CLG. Originally the CLG was going to be used to define the problems in the interface, but the author did not feel it was robust enough to define the inconsistencies noted by the subjects. The CLG was developed prior to the proliferation of the graphical user interface and does not make distinctions for some objects and inconsistencies specific to a GUI.

The six levels of Moran’s CLG were broken down into eight classes. The divisions were formulated by the author after looking at the types of inconsistencies found by subjects. Moran’s grammar did not directly apply to some inconsistencies, and the physical level was reworked to be appropriate for a direct manipulation interfaces. The author does not imply that this scheme is tested or validated, but it did provide a method for defining the inconsistencies in this experiment. Understanding Moran’s grammar and Robert’s (1988) example were very helpful for creating the classifications.

The bounds of each of these classes can overlap a great deal depending on the inconsistency. Note how the lexical level also includes changes in iconic representations. In the same manner that the word “Find” might be substituted for “Search” a different icon might be used to represent actions of similar meanings. This could also be classified as a semantic inconsistency. The distinction would come from how the subject phases their comments and the context in which it was made. For example, if the subject mentioned that the icons represent the wrong concept or metaphor then it would be defined in the semantic class, but if the subject only determined that two icons did the same function, or that two similar icons (like the phone icons) performed different function then it would be classified into the Lexical class.

Another example would be with the use of audio feedback. If a subject defined a beep as an inappropriate method for feedback then it would be classified at the semantic level, but if the subject determined that the beep sounded like an “error” beep when it should have sounded like a “good job – task completed” then it would be placed in the System State class.

No formal methods were employed for classifying the inconsistencies, but the author did complete the classifications twice to insure some uniformity in the classifications. Even so, others could interpret the inconsistencies differently and without a true taxonomy it becomes difficult to make consistent judgements.

4.3.2 Classes Used to Describe Inconsistencies found in the Personal Organizer

Physical Location (P) - This is equivalent to the Spatial level defined by Moran. Inconsistencies in the variation in the location of an object would be included. It also could include the relative positioning of menu choices, or how they were grouped (or separated), similar objects having different locations, or different objects having similar locations.

Memory (M) - When the program does not “remember” an event, object position, system state, or other activity, that a subject expects to be remembered, it was classified as a memory inconsistency. Other memory inconsistencies could occur with; window locations, previous search strings, and the storage of entered keystrokes or mouse clicks.

Semantic (S) - The semantic level could included the memory group in the CLG, but also would include items that had different meaning for similar words (icons) or different words (icons) with similar meanings. If the current system state would imply that a specific function, or option of a function is available, or not available, then these changes would be considered semantic inconsistencies. This level enumerates the concepts and not how to access them in the application.

Syntactic (Y) - The syntactic level includes high level objectives, methods for entering a command or initiating an action, and shortcuts. When items were presented, and when one method for initiating a command is available or unavailable also could be defined as having syntactic inconsistencies. The level specifies how the system operations get invoked.

Virtual Device (D) - Device refers to the method for activation or usage, and the use of a mouse, trackball, or keyboard. In this experiment it refers to the system objects that a user acts on, like a scrolling window, scrolling list, pull down menu, hierarchical menu, check box or dialog box. Users created

changes in the state of the interface with the “virtual devices”. This level specifies the appropriateness of the device and now how it is operated.

System State, or Feedback (F)- Any feedback that gives the user some information about the current state of the system, or what state the system would be in if an action was initiated. The arrows used to indicate to the user that a pull down menu was available would be in this level. Changes in the cursor, other visual feedback, audio feedback, window boxes, or other properties that make the user aware of the current system status were included in this level. Changes in the graphic layout of the screens would also be included. Removing the “look” of a Roledex card would be included in this level.

Lexical (L) - This level includes all changes in the wording or representation of any of the applications’ language. This also includes pictographic (iconic) representations. This could include error messages, prompts, and icons used to represent system properties or functions. Abbreviations, capitalization, and the type of language are part of this level.

Task (T) - (unavailable functions) - This contains system features that were not implemented in the application, but should have been available to the user (actual missing functions and not functions available and simply not discovered by the user).[5] Unavailable functions were “equally” unavailable in all versions.

Since these categories do not represent a true taxonomy it was difficult to consistently separate items . How the inconsistency was worded or phrased can change how it was categorized. For example, a discussion about the Sort feature could reveal inconsistencies at many levels (Table 35). Some parts of the table are fictitious to illustrate potential problems at all levels.

The inconsistencies identified in the twenty-two questions (22 topics) of the post experiment questionnaire are presented. These tables do not account for

---

[5] Any property, action, or function which was available, but not discovered by the user would probably be an indication of an inconsistency at one of the other levels.

the number of identical inconsistencies found by more than one subject in each topic. Table 36 lists the total number of inconsistencies identified by subjects in ascending order for each version. These data was treated to a Mann Whitney U test (Table 37). Subjects in Group 2 and 3 found significantly more inconsistencies than subjects in Group 1. Inconsistencies for each subject were ordered to approximate a normal distribution. Figure 10 illustrates the trends of each group. The graph not only shows subjects in Groups 2 and 3 had more usable comments, but that each subject in Group 2 and 3 had more responses than subjects in Group 1. So subjects, in general, were more responsive, and it shows that the comments did not just come from a few select subjects.

TABLE 35. Classification of Comments of the Sort Function into Levels.

Comment	Discussion	Level
“The Sort function was in a different place on every card”	The location of the icon varied	Physical Location
“The Sort function should remember that I sorted by Company Name last time and it should pop-up that menu choice first”	As a pop-up menu, lists of options should be enumerated with the default selection first (the most recent selection)	Memory
“The Sort function did the same thing in every function but there was a different icon for it in each function”	Different icons for similar tasks	Semantic
“The Sort function used a pop-up menu in the Address Roledex, but a Dialog box in the Phone Directory”	Different methods for activating the feature	Syntactic
“The sort function should use a dialog box not a pop-up menu to choose how to sort”	The wrong type of virtual device was used	Virtual Device
“There was no way to tell if the sort feature was working or not, the cursor did not change and there was no beep when it finished”	No feedback to the user to describe what the system was doing	System State - Feedback
“The wording in the Sort Dialog box was in all caps it should be written normally”	All uppercase was inappropriate for this dialog	Lexical
“You should have been able to sort by zip code for mailing lists”	This was unavailable in the Sort function	Task - Unavail. function

TABLE 36. Summary of Inconsistencies Per Subject Within a Group

Subject w/in Group->	1	2	3	4	5	6	7	8	9	10	11	Total Inconsistencies
Group 1	8	7	6	6	3	3	3	2	2	2	0	Σ = 42
Group 2 (I-A)	21	18	14	13	12	8	7	5	5	5	0	Σ= 108
Group 3 (I-B)	20	15	13	13	13	12	10	10	8	8	6	Σ= 128

In Ascending Order- This table does not include unavailable Tasks (T's)

TABLE 37. Mann-Whitney U Comparisons Between Groups for the Number of Inconsistencies Identified in the Treatment Versions

Comparison	Σ Rank for Groups in comparison	*p* Value
Groups 1 vs. Groups 2	90.5 vs 162.5	.0176 *
Groups 2 vs. Groups 3	112 vs 141	.3383
Groups 1 vs. Groups 3	70 vs 183	.0002 *

* significant at $p < .05$ *p* values are corrected for ties

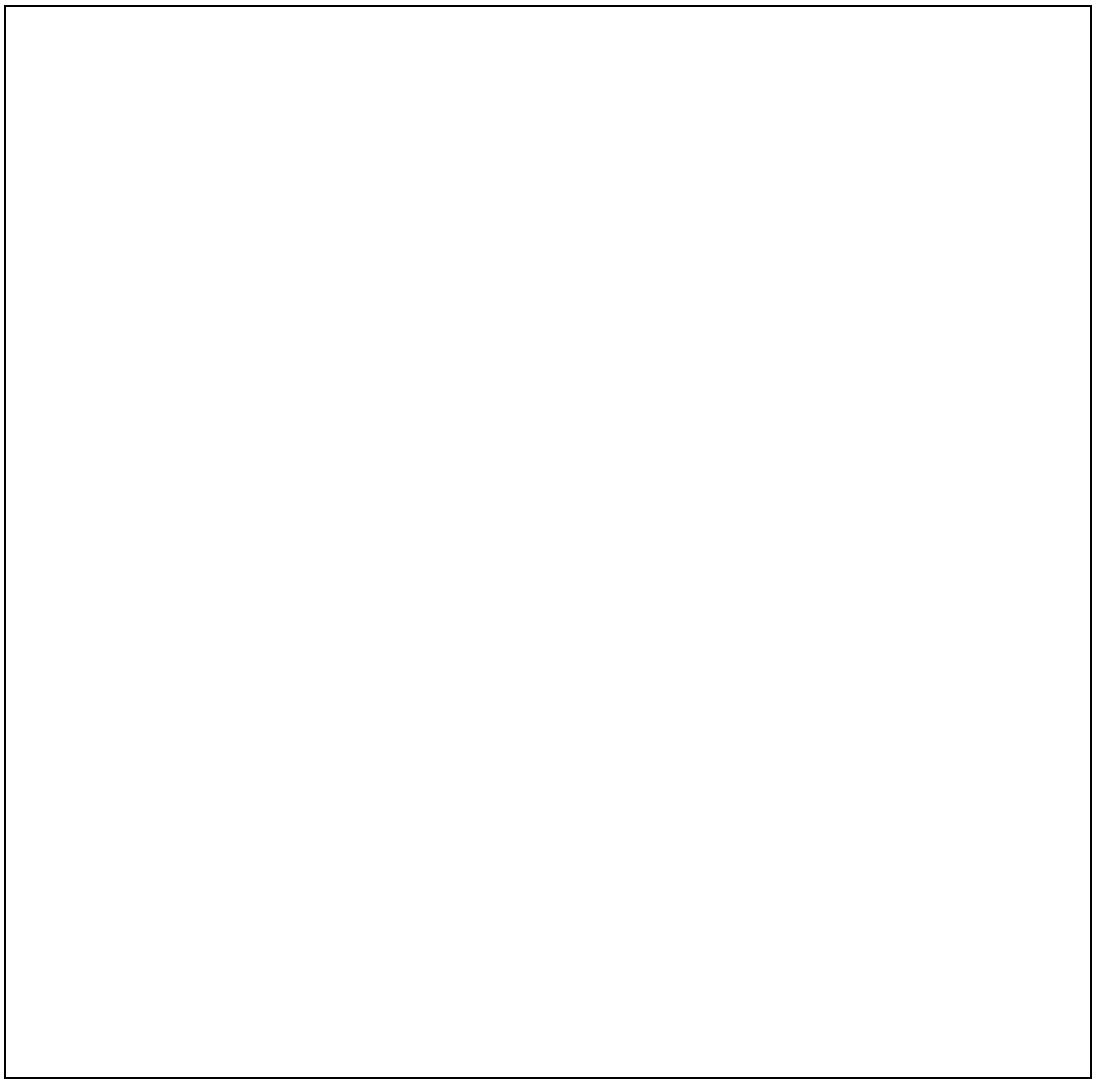


Figure 10. The number of inconsistencies reported by Subjects in Groups.

Table 38 outlines the summary of inconsistencies across the three groups of subjects. It contains the number of inconsistencies identified by Group for each of the twenty-two topics. For comparison the number of known inconsistencies is listed to the left of each group. Note that the number of inconsistencies seeded in the version used by subjects in Group 2 and 3 were almost identical. Under some topics the number of seeded inconsistencies was identical but the number recognized by subjects was different. For example in Topic 2 (Icon Placement) Group 1 found only 2 inconsistencies with none seeded, while Groups 2 and 3 found 5 and 10 respectively with the same number of seeded inconsistencies (9). One needs to recognize that the Seeded columns reflect the total number of *different* inconsistencies, and the Group columns reflect the *total* number identified. The total number identified might include numerous subjects finding the same inconsistency or each subject finding a different inconsistency. In addition, having defined the type of inconsistencies for each Topic in Appendix I-X does not preclude subjects from finding a completely different type of inconsistency without identifying the known inconsistency. The problem, as will be expanded on in the Discussion, was due to not having a true taxonomy for classifying inconsistencies. Without using multiple "experts" to classify the inconsistencies and to verify their place in a topic it is difficult to make judgements on the appropriate place for inconsistencies. Appendix I contains the raw data which outlines the specific type of inconsistencies mentioned for each question and Group.

Table 39 displays the data parsed by the type of inconsistencies identified by subjects and seeded by the experimenter. The number of location, syntactic, semantic, and feedback inconsistencies seeded in the two inconsistent versions are large and approximately equal. Group 3 found more semantic (7), device problems (7), and location (9) inconsistencies then the subjects in Group 2. These differences account for 22 of the 23 inconsistencies that separate the two groups.

TABLE 38. Summary of Inconsistencies Per Group

	Group 1			Group 2			Group 3		
Topic	Total Found	Unique Found	Seeded	Total Found	Unique Found	Seeded	Total Found	Unique Found	Seeded
1- Menu Structure	3	2	0	11	10	7	11	9	6
2- Icon Placement	2	2	0	5	4	9	10	7	9
3- Edit Keys	1	1	0	2	2	3	3	2	3
4- Menu Item Order	2	1	2	7	5	2	11	8	6
5- Help System	4	4	4	8	6	7	6	6	8
6- Navigation Arrows	1	1	0	3	2	2	6	2	2
7- Icons for Function	4	3	1	3	3	2	4	4	2
8- Find within Function	3	3	1	6	1	2	5	1	2
9- Find between Functions	2	2	0	4	1	1	4	1	1
10- Function Key Layout	2	2	0	7	4	1	3	3	1
11- Phone Setup	1	1	0	8	5	2	7	4	2
12- Dialing a Number	3	3	0	2	2	1	5	5	1
13- Error and Other Messages	4	3	0	2	2	0	4	4	0
14- Auditory Feedback	3	3	0	4	4	2	3	1	2
15- Visual Feedback	1	1	0	4	3	4	3	3	4
16- Clock Function	0	0	0	3	3	1	3	2	1
17- Wording	0	0	1	7	3	5	7	4	3
18- Using Tab Keys to Nav.	1	1	0	2	1	3	7	2	3
19- Color	0	0	0	1	1	1	3	3	1
20- Floating Palettes	0	0	0	9	9	3	8	3	3
21- Function Specific Actions	1	1	0	2	1	1	7	3	1
22- Pop-up Menus/Icons	4	4	0	8	6	1	7	5	1
Totals	42	38	10	108	78	60	128	82	62

Each left/right set of cells contains the number of inconsistencies found in each version of the interface and the number of seeded inconsistencies known to exists prior to the experiment. Unavailable Tasks are not included in this table.

TABLE 39. Number of Found and Seeded Inconsistencies per Group and Type

	Group 1			Group 2			Group 3		
Type	Total Found	Unique Found	Seeded	Total Found	Unique Found	Seeded	Total Found	Unique Found	Seeded
Physical Location	5	4	1	35	28	18	44	29	18
Memory	2	2	1	13	4	5	9	2	5
Syntactic	6	4	2	11	5	11	19	12	10
Semantic	4	5	5	5	9	10	12	12	9
Virtual Device	10	10	0	16	12	5	23	13	5
System Status (Feedback)	10	9	1	14	11	8	7	4	12
Lexical	5	4	0	14	9	5	14	10	3
Unavailable Task	15	13	2	16	10	2	24	18	2

Each triad of cells contains the total number of inconsistencies found, the total number of unique inconsistencies, and the number of unique inconsistencies seeded into the versions presented to the Group in Set 2.

Figure 11 helps illustrates where subjects found inconsistencies in the interfaces. Clearly subjects had an easier time finding physical location inconsistencies in the interface. Figure 12 shows that the number of physical location inconsistencies seeded in the three versions of the application was also higher than the other types. One can also see that for many of the other types of inconsistencies Group 3 found more inconsistencies in their version of the interface than subject in Group 2, even though Figure 12 illustrates that the seeded number of inconsistencies was almost identical.

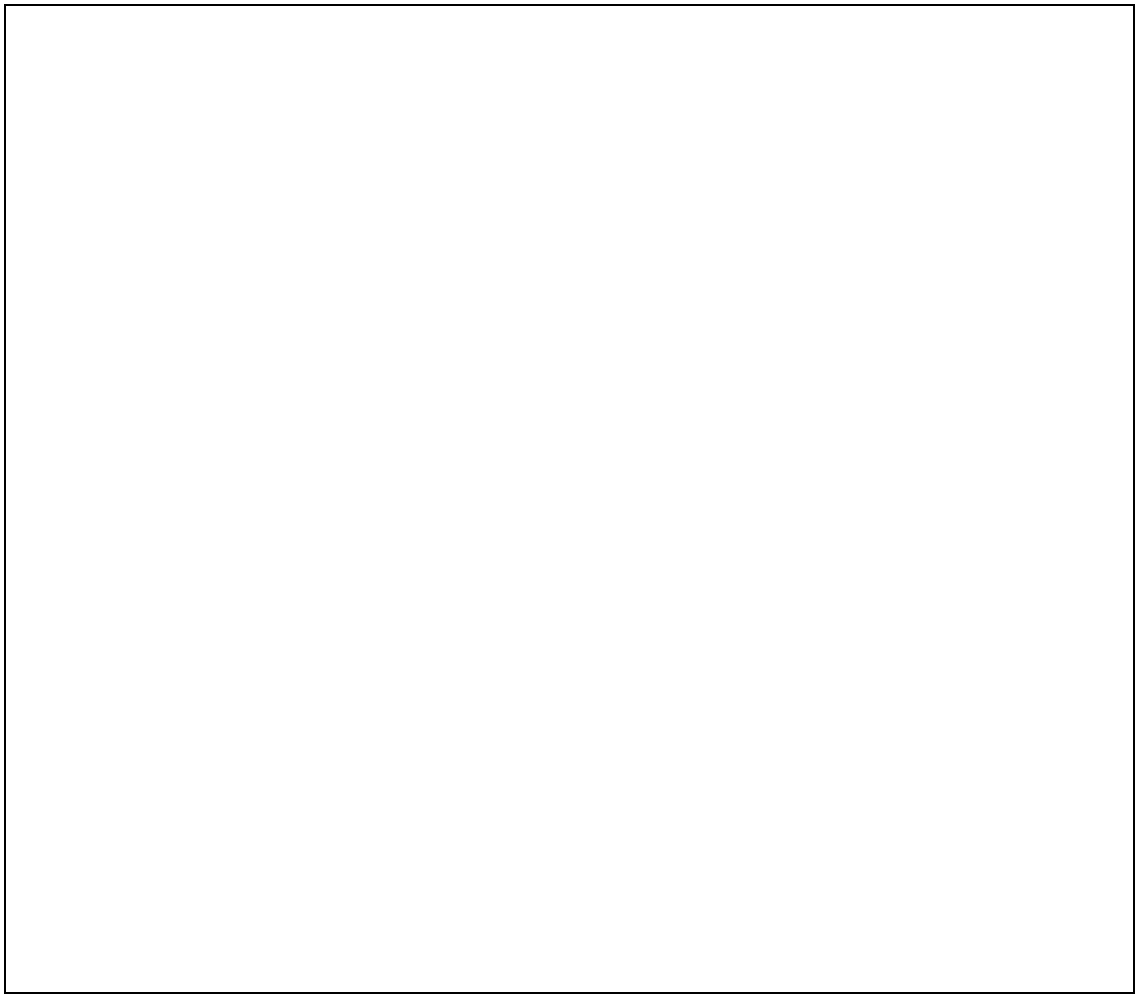


Figure 11. Comparison of the types of inconsistencies identified by Groups.

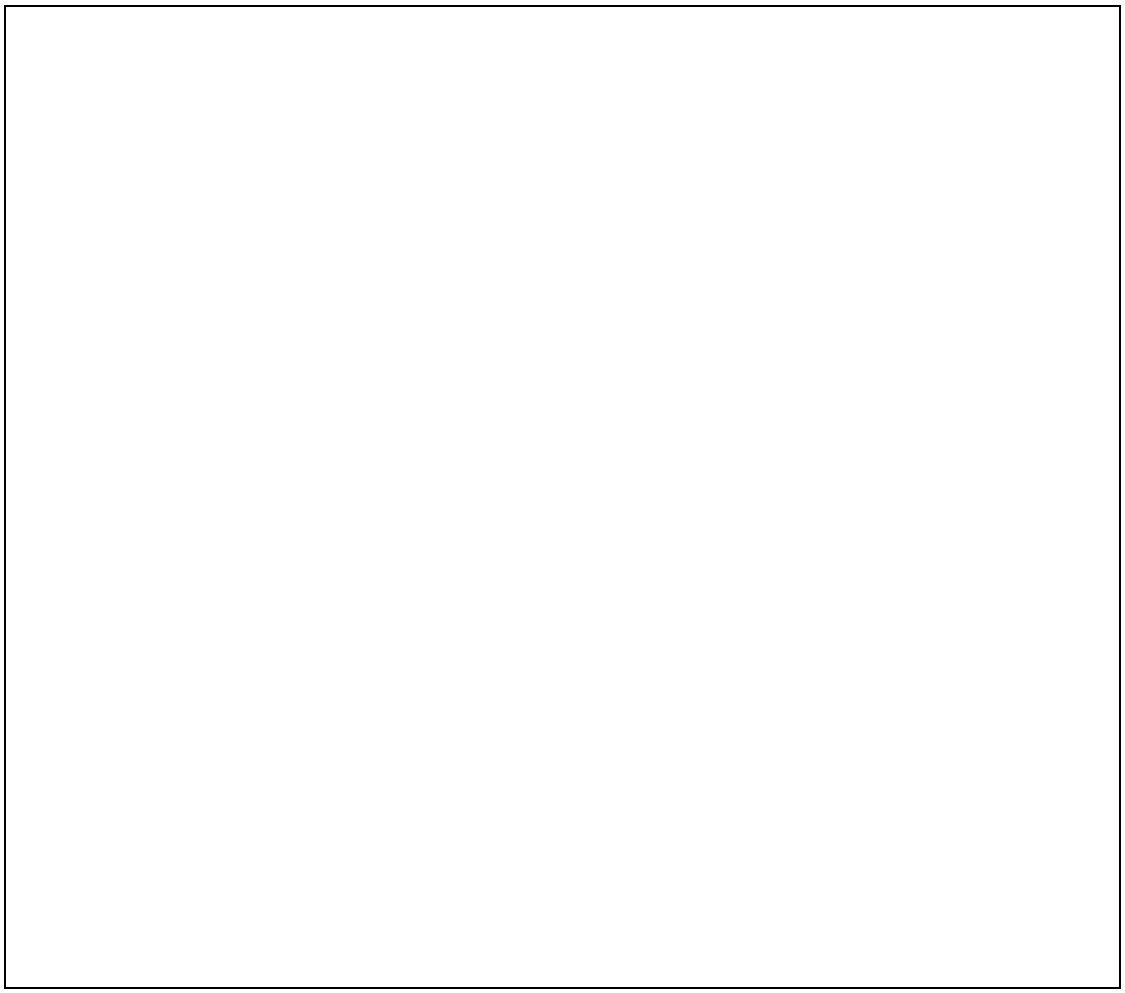


Figure 12. Comparison of the types of seeded inconsistencies identified by Groups.

CHAPTER 5 DISCUSSION

Users play a significant role in the development of graphical user interface design. Since unlimited access to end-users is not feasible, the time taken for their inputs should be optimized. If designers understand what types of inconsistencies have the most significant effects on performance, then the time spent in creating the interface will be optimized.

The ability to determine if specific categories of inconsistencies can be classified as better or worse than others will allow designers to concentrate their efforts toward the type that will cause the most problems with user performance. This in turn will allow users in the design loop to concentrate on areas of the interface which need attention. Increases in user acceptance and productivity should result.

In this research an inconsistency could appear in more than one level. For example, a syntactic inconsistency could impact the within application level and the platform level. Each having a different affect on user performance. This research concentrated on inconsistencies in Level 1 and 2 (as defined by Robert, 1988), but revealed potential problems that affect other levels of the the hierarchy.

The ANOVA with time as a dependant measure in conjunction with the PI revealed one of the most important findings; objective and subjective measures do not always concur. Neither method alone was powerful enough to reveal differences between UID's. The time measure showed inconsistent version - B (the treatment for Group 3) as having the most impact on objective performance and the subjective PI showed that subjects in Group 2 liked inconsistent version - A the least. Users think that because an interface is inconsistent that they will not slowed by the interface. In this case, users perceived that the time to complete tasks was affected, when in fact it was not significantly affected.

5.1 Objective Performance Measures

It was surprising to find that the step data was not revealing. It was hypothesized that the use of the menu items with the mouse would decrease as the subjects understanding of the interface increased to the point that the quick keys would be used more frequently. This was not the case, as the only change during the sessions was the decrease in the number of icon clicks. As subjects improved they required less steps to complete the tasks (i.e., deceased their error rates). This was accounted for by the decrease in the number of icon clicks. The menu was almost never used by subjects. Most subjects used the quick keys for the Edit commands and many only went to the menus to choose the size for text. Since the Sizes could not be selected any other way the subjects were forced to use the menus for this task. It was undermined if or how long would be required for subjects to change from predominantly mouse click and menu usage to quick keys. The time for this conversion is probably due to the complexity of the design, the mnemonics used for functions, and the frequency of the items use in the application.

The correlation between the time and step data (.807, $p < .0001$) was high. The uncorrelated part can probably be attributed to the subjects strategy in performing the tasks. From observation it was clear that some subjects took the "think and act" method that increased the time spent on a task while minimizing the number of steps attempted. Others used the "brute force" method where subjects click many buttons until they found the one that worked. This led to an increase the number of clicks, but appeared to be the fastest method for subjects to work. This method exposed the subject to other parts of the interface which could be of use in future tasks. This is similar to the game of Concentration where a subject will "overturn" an item that could not be "matched" with the current task, but could be used at a later time to complete a different task. This comes from observational data and since subjects were not categorized my these methods it is now known if the method significantly affected step data.

The ANOVA on time showed that Group 2 was hindered by the inconsistent application when performing the first Block in the treatment condition. The subjects did recover quickly and performed as well as subjects in the other two groups in the second block of tasks in Set 2. This probably implies that subjects

are robust to interface inconsistency. This finding was similar to what was reported by Kellogg (1988). It could reflect the limited nature of the inconsistencies, but this is not substantiated. Subjects in both groups found the inconsistencies a problem, as will be shown later in the discussion of the subjective ratings. Visually the version administered to Group 3 was the most distinct (Appendix J), it also contained completely different rules for using the navigation features, the menus, pop-up icons, and many other features which were different from both the consistent version and the inconsistent version used by subjects in Group 2. With all these differences the subjects performance was still as good as Group 1. In addition, Group 2 and 3 had no trouble returning to the control version of the interface. Figure 6 shows that Group 3's performance was better than subjects in the control version, but these differences are not significant. By the last block all three groups were within a few seconds of each other. Although these data does not help understand long term effects, the short term effects of these inconsistencies was clear, subjects were robust in their ability to give correct responses to identical stimuli in different surroundings. One explanation seems to encapsulate the reason for Group 3's good performance; the localization of consistency.

The interface used by subjects in Group 3 was defined as "inconsistent" to the control interface and the Macintosh system. This "other" version was outside of the subjects local concern. The transfer of knowledge from the control version (outside of the subjects localization of consistency) would not have as much impact as knowledge gained from well known applications (long-term retention and transfer that is difficult to extinguish), or even from using the inconsistent version in the first few minutes of Set 2 (short-term memory from local events). For example, this program used a navigation palette, which by itself contributed to a variety of inconsistencies. The navigation palette is similar in many regards to tool palettes in graphics, page layout, and other applications. This might help a user to understand its function and accept the palette as "consistent" with what the user had remembered from long-term usage patterns. Those memories would not contain as much information of the control version, which was used recently, but not extensively (i.e., not enough experiences have been obtained to give the user long-term knowledge of the control version. A user might find that the palette makes sense in the context of the application and would consider it

consistent within the application. This makes the palette consistent within this version of the application and consistent with other completely different application, but inconsistent with the other version of this application! Thus consistency applies at two levels, but not an inclusive level. This can have profound impact on how the application is viewed. A general system object (the palette) is consistent with other applications, while the features on the palette (the application dependant functions) are consistent within the context of the inconsistent version. One needs to be able to discriminate objects so that they can be properly described at a given level. Once described they can be compared to definitions form other objects in that level to look for a common thread. This thread could be the link that creates consistency from the users point of view.

The ANOVA on time also showed that subjects in all three groups learned how to use the application with little effort. Performance time decreased by a factor of at least two and sometimes three for many subjects. The average times decreased from in the 20 to 25 minute range, with high variance between subjects, to about 11 minutes, with most subjects completing the tasks in the 10 to 12 minute range. Figure 13 shows the standard deviation of groups decreased dramatically. By the last block subjects had about the same level of performance independent of the treatment condition. This implies that there was no lasting effect from moving from a consistent to inconsistent and back to a consistent version of the application. This type of data is useful in determining the level of flexibility designers should offer end-users. This shows experienced subjects were robust to changes in the style of interaction, even within the same application, and were more inclined to perform better when the interface style was completely different.

Programs like Word 4.0 and Nisus 3.0 give the user flexibility in creating personal menu structures. Adding a powerful macro making capability, and a utility like Icon It 2.0, for creating a customized icon palette, the user interface can be made to appear completely different. With data showing that users can move between functionally similar interfaces with distinct user interfaces means that designers can let the user determine what is necessary. Unfortunately, many users are resistant to change and might not be as inviting to change as experienced Macintosh users in a controlled laboratory experiment. Novice

users would probably fair much worse and would not have the skills to learn this interface as fast. In addition, the subjective measures, to be discussed next, showed that subjects were well aware of the changes and although the objective measures did not show differences the outcome from the subjective measures was more distinct.

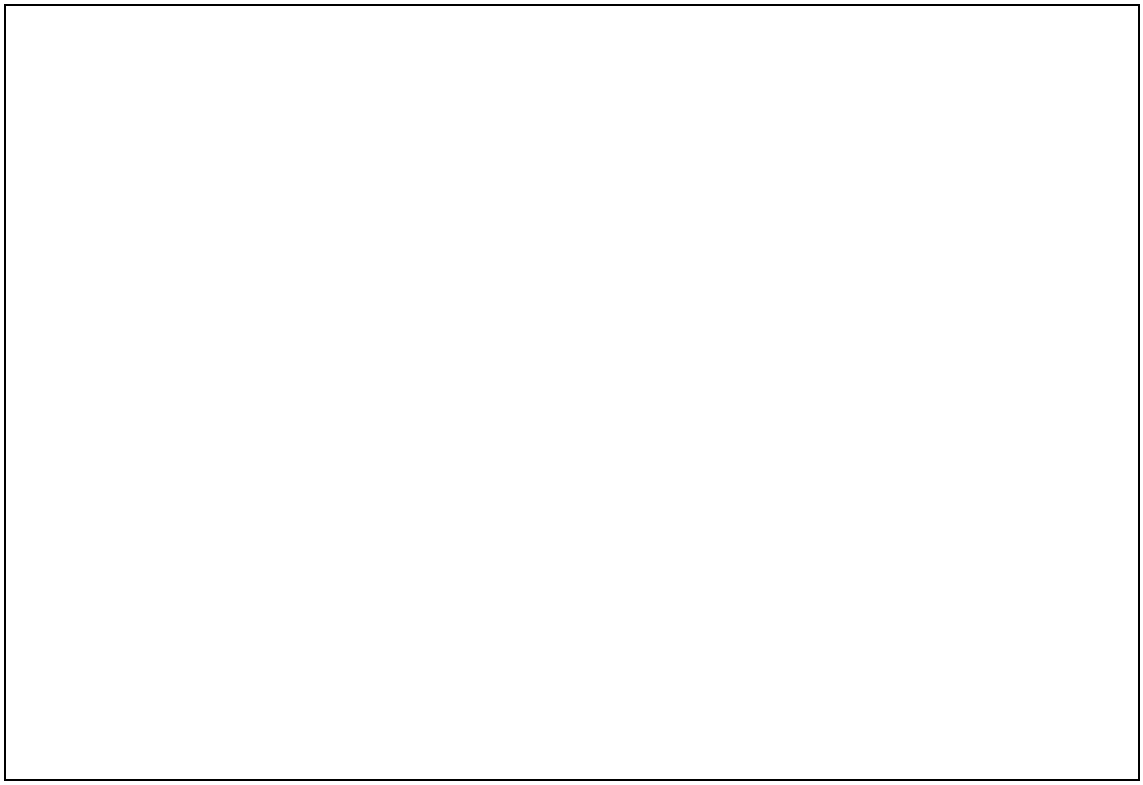


Figure 13. Deviation in group scores across Blocks.

5.2 Subjective Performance Measures (Discrete Response)

5.2.1 Preference Index

The PI served two purposes. It allowed one to gage subjects opinion of the user interface. And it allowed one to formulate a set of adjectives that can be

useful in describing consistency in UID. If a designers keeps asking a user if the interface was “consistent” the word will start to loose any useful meaning. User might be more comfortable using words like pleasing, cooperative, dependable, simple, interpretable, maintainable, easy to learn, not frustrating, and productive. The PI also found that the interface made the user feel optimistic, competent, and pleased. Although the questions were posed in referencing the “application” and the “interface” there was no analysis as to if subjects viewed these as relating to different principles (Appendix D). References to the “application” might imply the engine, or the algorithms, used to feed information to the interface.

Although every application includes an interface and an engine, most users only interact with the interface. This makes it difficult to pass a judgement on the application engine without an intimate knowledge of how the information is processed. So, it was assumed that subjects perceived the application as the interface, except for when functional aspects were considered. The adjective pairs that were correlated with the Consistent-Inconsistent pair showed that subjects rated these attributes in a similar manner to the C-I item. During the experiment some subjects expressed confusion to the experimenter on how to answer the Adaptive, Safe, and Maintainable items. All three turned out to not be correlated with the C- I item. A factor analysis would reveal combinations of items that were related, but since the C-I was the focus of the PI, this was not done. A factor analysis would have shown what other items were related, but any factors without the C-I item would not be of interest.

Redundant-Concise was difficult to interpret. “Good”, ”Consistent”, or “Dependable” are all positive traits but it was difficult to which adjective was the positive trait in a computer interface. Concise usually implies straight to the point, but a program is considered robust when it gives the user alternatives to imputing an action. For example, the Word 4.0 rules allows on to set the margins for a paragraph by directly manipulating the ruler icon. One can also double click on the icon, use a menu, a quick key, or a function key to bring up a dialog box for typing in the measurements. This redundancy is generally well received in the computing environment, but it can also cause problems.

In a standard programming language all the code resides in a single “document”. It might be parsed with subroutines, and function call, but it is all

contained as a single list. Search and replace functions are easily implemented and moving, cutting and pasting are all easily accomplished. This single document is concise. In Hypertalk programs code can reside in buttons, fields, or other objects on the card,background,window, program, or even in other Hypercard stacks or external functions. These are over one dozen distinct locations for code. Controlling and organizing the code can be very difficult when hundreds of buttons, fields, and objects are used. Novice programmers tend to write code in each object which becomes redundant when much of the code could be written at a higher level in the message hierarchy.[6] In most cases redundancy in Hypertalk should be avoided, but in the interface it is appreciated. Another aspect of redundancy concerns identical information residing in more than one location. This could be considered a good trait when it is used in more than once place, but bad if it is difficult to update all occurance of that information. A global search and replace would bve useful, but being able to link the information so that updates in one location will be reflected in other locations.

The PI showed that subjects were clearly aware of the inconsistencies in the interface. An ANOVA on the PI scores showed that Subjects in Group 2 and 3 rated the interface as worse than the consistent version. Subjects in Group 3, the group who's timed performance was not significantly different in Set 2, found their interface relatively worse than subjects in Group 2. All Groups rated the consistent version similarly in both Set 1 and 3 which implies that the differences noted by subjects came from the same perceptual basis. Subjects in Group 1 were very consistent in their ratings. Even though the subjects used the same interface three times, their PI scores did not indicate any differences. This can be used as a measure for boredom. Under the assumption that if subjects were getting bored then it would be from having identical interface sin all three sets. Since they were consistent in their ratings one can judge that subjects were not

---

[6] Hypertalk based languages use a message hierarchy to pass information from one object to another. A piece of information goes "up" the hierarchy from buttons, fields, and objects to cards, backgrounds, windows, and stacks. It is possible to write one piece of code in the higher levels (background or window) that can be called by other objects. This can reduce writing redundant code in many similar objects.

biased by having multiple measures on similar interfaces, and that they were consistent with how they rated the interface.

If subjects in the treatment version were “relieved” to have the consistent version in the third set then that might influence their ratings. Although not significant ($p < .05$) subjects in Group 1 showed a slight decrease across sets, while the ratings for both Groups 2 and 3 increased from pre-treatment levels. Although, this experiment could not confirm any halo effects, maturation effects could occur when evaluating new applications. If a subject already had been used in evaluating an interface at an early stage of development. and found it very poor, they might overcompensate there estimation of an improved version of the interface. Subjects might find it vastly superior to the old version, while users who had not seen it might be only moderately impressed since they do not understand how poor it was previously.

Many designers might have limited resources to users, and if halo effects existed then designers would have be careful to use subjects that were not exposed to the interface previously. The negative effect would entail not giving the user enough exposures to the interface to make detailed observations. There is a trade-off between using a few subjects for long periods and more subjects for less time. The former allows for a more detailed analysis of the interface, but keeps the designer from determining how the interface works at a given stage (since existing users are biased). The latter allows one to be aware of how users will relate to seeing the interface for the first time (first impressions are always important), but will keep the designer from understanding potential low level conflicts and problems in the functionality. A combination of both would help to alleviate some of each of these problems, but determining the right mix for a given application is unknown.

5.2.2 Rating Twenty-Two Interface Elements as Consistent of Inconsistent

Subjects were asked to rate twenty-two parts of the interface as either consistent or inconsistent. The Mann Whitney U test showed that Groups 2 and 3 found more parts to contain inconsistences than subjects in the control groups. This shows that subjects could identify specific areas as consistent or inconsistent. It could be helpful to gather these ratings to get a general

understanding of where problems in the design are occurring. It does not reflect the difficulty in determining what in the part of the interface was inconsistent. Accordingly this type of rating would only be good in the early stages of development to confirm that certain parts of the interface would require more effort than others. This can be especially useful in large projects where different groups are working on distinct parts of the interface. Subjects could be used to identify if "general" parts of an interface as consistent. Further analysis could be used to identify what kind of inconsistency were affecting the subjects.

5.2.3 Comparing Actual and Estimates of Completion Time

After the subjects completed the experiment they were asked to write down how long they thought it took them to complete each block. What was surprising was that subjects in Group 2 and 3 were more accurate than the subjects in the control version. The subjects in the control version always could see the time in the top right corner of the screen. The time presented on the Weekly Planner and Calendar in the inconsistent versions. The time was also displayed in the navigation palette used by Group 3. Since the ability to actually tell time favors subjects in the control version, the repetitive nature of the tasks, probably caused the used to loose track of the time. Subjects who used the inconsistent versions were able to keep a better track of the time. Losing track of time while working on a computer is not unusual, but that the type of interface affected the perception of time is important. It is useful to know if a subject thinks it only takes "10 minutes" to complete a task when the actual time was 20 minutes. This would be a positive trait of the interface like when watching a really good movie and it seemingly ending before one expects. Getting "caught up" in using the interface can be a good trait for an interface, unless one loses track of time in a deadline situation. In this case, it reflects that change in the interface style can cause the perception of time to vary. Perception of time can be more important to the designer than actual time. As discussed in Section 5.4, if one can make a user feel that the system is performing quickly then it could mask if the application is actually not designed properly. This might be good for the developer in the short run, but bad for the user in the long run. As discussed in

Section 5.4, the perception of time can affect the users decision to continue to use the product.

The correlations of actual to estimated tome do not reflect if subjects under or over rated their times to compete the blocks. Subjects in Group 1 and 2 overestimated the time to complete the tasks in 28 and 32 of the 66 estimates, respectively. Subjects in Group 3 overestimated the time in 42 of the 66 estimates. There did not appear to be any pattern (i.e., most overestimates do not come from a particular Set), but again this measures shows that subjects in Group 3 did experience something that changed the way they perceived the inconsistent version. The version used by Group 3 was found to be very dissimilar to the other two versions, and subjects might have perceived that the dissimilarities were causing their time to complete the tasks to rise in the second and third Set. Subjects anticipated that changing the interface would impact their performance more than subjects in the other two conditions.

5.2.4 Direct Comparison Questions

This data mimics the findings shown in Figure 8. The differences found in the PI scores are mostly reflected in these results. The only difference is in the PI where differences between Group 2 and 3 were significantly different for Set 2. That comparison is similar to Questions 1 and 2 for the Group 2 & 3 comparison. Unfortunately, this analysis was not able to uncover a difference between Group 2 and 3. It is unusual that subjective opinion actually favors the version used by subjects in Group 3 and that the subjective measures cannot discriminate it from the version used by subjects in Group 2, which was similar to the control version in many ways but universally rejected by users. The possibility of the similarity between the control and the version used by Group 2 as for more problems is discussed elsewhere (Section 5.3.2).

5.2.5 Comparison Questions Between Versions in Sets Two and Three

These questions determined that subjects in Group 3 did not feel that the consistent version was not significantly different than the inconsistent version using five adjectives pairs. The other measures using between group analysis revealed that there was a distinct difference in subject ability to recognize that the interfaces were different, but this measure reveals that they did not like it any worse because of those differences. The items used to make up the Direct Comparison Index implies that subjects in Group 3 did not find the interface in the control version any better, more understandable, more consistent, easier, or simpler than the treatment version.

The subjects in Group 2 found their version objectionable as compared to both Group 1 and 3. So, this implies that subjective opinion from Group 2 was unfavorable and both the control and version used by Group 3 were aptly described by the positive attributes in the index. Thus, subjects could discriminate between what was inconsistent and what was different but consistent. Group 3's description of the treatment version as consistent, when Group 2 defined that version as inconsistent implies that the level that consistency was defined was inappropriate for these applications. Many of the attributes that would make the treatment version used by subjects in Group 3

consistent are defined at the lowest range of consistency (Robert, 1988). This is discussed in further detail in Section 5.3

5.3 Subjective Questionnaire (Free Response)

5.3.1 Comments on Free Response Questions

Each subject made comments for the twenty-two topics in the post-test questionnaire. It was predicted that subjects would be more responsive to the questions when they found fault in the interface, than when they had something good to say. Subjects were asked to comment on the consistency or inconsistency of the interface. Subjects in Group 3 were significantly more responsive with their comments. Group 2 and 3 wrote 16% and 48% more text, but identified 145% and 198% more inconsistencies. These differences were significant (Table 37). Subjects might be aware that the intent of the experiment was to reveal inconsistencies, and thus were less inclined to make comments about the consistencies in the interface. If this was true then subjects should have written substantially less in the control group. If consistency had no effect on the amount of written comments then one would not expect such a large difference in the number of usable comments. Inconsistencies could have been easier to identify in the inconsistent versions since there were more potential problems. When compared to the seeded inconsistencies (Table 38) one sees that both inconsistent version had six time the number of seeded inconsistencies, yet Group 3 still found 22 more inconsistencies than subjects in Group 2 (Table 36). Although not statistically significant, it is practically significant. The identification of inconsistencies should always be treated as “significant” by designers. Just because “90 out of 100” users did not find an inconsistency (i.e., $p = .90$) it might be put aside. Since it was found, then there should be some effort to fix the problem. Fixing the problem might involve a minor change that could be incorporated without affecting other vital parts of the interface. On the other hand, it could impact another part of the interface and cause a more important part of the interface to become inconsistent. Although this research did not address how to determine what levels of inconsistencies cause the most harm to performance some suggestions are made in the next section.

Groups did not show the difference in the number of inconsistencies reported due to a few verbose subjects (Figure 10), but that subjects in all three groups contributed to the total number of inconsistencies found. As the graph shows some subjects found many more inconsistencies than others, but this was constant between groups. It would be helpful to know what characteristics were similar between subjects who found a high number of inconsistencies, and if anything could be said about the subjects who found few inconsistencies in the interfaces. Profiling users was not the intent of this research, but other research has looked at expert and novice behavior and other attributes which relate to user performance (Mayer, 1981; Allwood, 1986).

5.3.2 Levels Used to Describe Inconsistencies

Creating the eight levels for describing GUI is primarily based on Moran's (1981) work. Creating a classification scheme for designers to analyze a user interface, and creating a command language (primarily used by researchers) contains two contradictory objectives. Designers are concerned with applying existing guidelines, especially in the Macintosh environment, as opposed to researchers who want to built methodologies, analyze systems, and formulate hypotheses.

The classification scheme was difficult to apply. If multiple experts were available to rate inconsistencies, then they can each assign the inconsistencies separately and an overall measure can be used to define where to assign inconsistencies. Virzi (1990) used this approach to analyze interfaces that were also reviewed by users. A pool of experts was not available, and as Table 35 shows slight differences in how a subject can word comment about the Sort function could affect how the problem is classified and to what the solution is applied. Accordingly it would have been nice to use experts to clarify some comments. Each type of inconsistency might require a different type of expert. If comments applied to the Physical Layout and Lexical levels then a graphic and industrial designer would be needed. Comments classified in the Memory and Feedback level could be addressed by a programmer. The Semantic and Syntactic level might be fixed with the help of human factors engineers and experimental psychologists. Finally, inconsistencies in the Task level might be

addressed by a marketing specialist, who keeps up with competitive products. The marketing or competitive analysis could determine if the time and cost are justified in adding additional functionality or interface enhancements. Device inconsistencies should be addressed by research and could be applied across applications, where as the other problems might have to be solved at the application level. The problems in this level can be addressed in a similar fashion to the standard input device studies.

One area was found to be difficult to classify: graphics. When subjects mentioned the color or other graphics that were not directly involved in giving the user some feedback they were classified in the feedback level. For example, the graphics which created the “look” of an address roledex card competed the roledex metaphor by making the screens look like real roledex cards. The absence of these graphic cues would be classified as an inconsistency in the feedback level. When the subject found the color of the main screen inconsistent, there were no good categories to fit the inconsistency. Although the feedback level was used for this type of comment, their might be a need for a another level that would only address “cosmetic” issues.

Another problem in the classification scheme comes from how subjects interpreted the twenty-two topics. If a subjects though of the topic in general terms then they would tend to add comments that were related, but not exactly covered by the topic. Many of the seeded inconsistencies did not exactly fit into any of the topics. Instead of not using them and making the interface look more consistent than it actually was, the seeded inconsistencies were placed into the topic that would be closest to a match. One cannot directly compare the number of seeded and found inconsistencies due to two main issues; 1) the number of seeded inconsistencies represents the number of *different* inconsistencies, and the found inconsistencies might contain more than one subject mentioning the same issue, 2) The seeded inconsistency might not be noticed by the subjects while another inconsistency not documented by the experimenter might be identified by the subjects. It should be more important to see how many inconsistences used can find than how many different inconsistencies are found. The frequency between subjects is more important than the number of distinct inconsistencies. One can always probe into the frequency data to reveal distinct inconsistencies, but this is a tedious and difficult task without a proper taxonomy.

Determining when two users have said the same thing is more difficult and time consuming than determining if the comments can be categorized in the same level. Unfortunately, only the number if *distinct* inconsistencies found be subjects can be directly compared to the number of seeded inconsistencies.

It appears that "obvious" inconsistencies predominate the type identified by subjects. Icon placement, Find function memory problems, pull down and pop-up menu issues accounted for many of the found inconsistencies. These are also the ones that users would be the most familiar with when comparing them to other Macintosh applications. Although users rarely used the menus via the mouse, hence rarely "seeing" the menus when using the application, subjects easily identified problems with the menu's structure, use, locations, and lexicon. Experienced computer users in the collegiate environment are probably exposed to a wider variety of software than a typical end user in the office environment. The exception being the system operators, computer hackers, and the resident computer "experts". Users frequently scanned the menus to see what features were available, but then used the quick keys or icons to actually perform the operations. Many icons did not have corresponding icons (some had quick keys), while many menus did not have icons (but might have quick keys). The availability of each method for a specific feature did not change between versions. For example, the Return to Opening icon and function key were available in all three versions of the application. It did not have a quick key or menu choice in any version. The theory that eighty percent of the problems can be found in 20 percent of the time probably applies more to these type of visually distinct inconsistencies than to the lower level semantic and feedback issues.

Subjects in Group 3 found twice as many problems in the Semantic (e.g., not having the delete feature available from the menu in all functions) and Device level (e.g., dialog boxes to change the Phone Setup parameters) than subjects in Group 2. Both inconsistent versions were seeded with the same number of inconsistencies, and were basically equally appropriated among the twenty-two topics. So, why did the subjects in Group 3 find so many more? The answer probably comes from the definition of inconsistencies. Seeded inconsistencies were defined relative to the control version of the application. The navigation palette used by subjects in Group 3 was defined as inconsistent to the icon bar used in the control version. This inconsistency across applications must be

compared to the subjects view that the palette was consistent within the context of the current version of the application. Users centered on the consistency within the application as opposed to across applications. If the quick keys for the edit commands were changed (X,C,V) then subjects would probably be very adamant about inconsistencies with other applications. Since this palette would have no function in other applications they confined their review of it within the application.

Looking at the inconsistencies found in the Function Key Layout in Table 38 shows that subjects in Group 2 mentioned 9 inconsistencies. The experiment only seeded one inconsistency. In this case the one inconsistency (ordering function keys alphabetically) was identified by many users. The standard function key placement uses F1 to F4 for the Undo, Cut, Copy, and Paste edit keys. No subject identified that the keys were ordered alphabetically. The keys were ordered with a less appropriate strategy in the version used by Group 3 (ordered by most used functions and most used feature within that function) (See Appendix I - Topic 10). This strategy made for an unusual order but only 4 comments were made by subjects in Group 3. The difference probably is due how the order of the edit keys was identified. In the version used by Group 2 the edit keys used very dissimilar keys (F3, F4, F8, and F13) and Group 3's used keys similar to the standard layout (F1, F2, F3, F14). The order used in Group 3's version was not consistent with the standard layout, but if subjects did not use the keys frequently they perceive them as being placed in the right place. It was more apparent to users in Group 2 that F8 and F13 were definitely not the correct place for the edit keys. Although these were all experienced subjects none mentioned anything about the layout similarity between the F1 to F4 keys and the Z/X/C/V quick keys.[7] In this case the similarity of the standard and inconsistent design in version B (used by subjects in Group 2) masked the inconsistency. This type of inconsistency would become more apparent to users

[7] The order of the edit keys is identical for the function keys and command keys when properly designed. The F1 (Undo), F2 (Cut), F3 (Copy), F4 (Paste) layout is identical to the Z (Undo), X (Cut), C (Copy), V (Paste) layout in the bottom left corner of a QWERTY keyboard.

when working with multiple programs where one frequently uses function keys. With the menu bar, the quick keys, and sometimes icon equivalents for the edit keys, the addition of the function keys might be too redundant for experienced users. If properly labeled the function keys can provide novice users quick access to 15 commands with just a single keypress. When the function keys are used with the command, option or shift keys then they can become as esoteric as many of the quick keys found in complex applications.

Overall, the summary of inconsistencies does one little good without the original comments. A designer would be best to have the comments summarized into groups of specific statements that apply to a topic. Unfortunately, this requires a great deal of time and patience to go through and analyze the comments. Collapsing the comments into groups of identical comments, and then discussing each one, would require a very lengthy discussion. If one did do this they would probably find that observations taken during the session would reveal the same information. During this experiment the researcher created ideas about how user manipulated the interface. Although running totals of the number of subjects who supported each idea was not recorded, it does give a general idea of how users worked with the interface. As more users were administered the experiment it became more apparent which of these theories were being applied by a large portion of the subjects. This type of observational information can sometimes be enough to keep designers busy in their effort to improve a UID. When more specific data would be required a VCR tape can be used to record the sessions. Once the theories are formulated during the original sessions, the list of theories can be used to review the tapes and come up with a count of the number of subjects who's actions would support the theories. Subjects could then be brought back in another session to rank the problems. This was done by Virzi (1990) with some good results. He concluded that the experimenter only had a moderate degree of agreement with subjects on defining the impact of usability from the known problems. This agrees with the theory that subjects are effective in identifying problems in the interface (Robert, 1988). The theories formulated during this experiment are presented in the next section.

5.4 Observational Data

During the experiment the experimenter logged over one hundred and twenty hours of observation time. This does not include one hundred hours for observing pretest subjects. During this time the author formulated ideas as potential guidelines, and rules which subjects seemed to follow. Help systems, navigation strategies, messaging and other areas have been covered in research literature, these and other topics discussed herein have not been applied to GUI's in the more useful form of guidelines, rules, or style guides. The Apple guidelines (1987)[8] did not address many of the issues, but were found to contain some important ideas discussed. Some were discussed in a different context, and many developers only have the guidelines and other available software to guide them[9], and the body of scientific research would be unknown to the developer. Most designers only have time to apply the rules, and not to review the research that established the rule. Since the emphasis of this research was to look at the effect of inconsistencies on the entire interface, and not just on one aspect, the discussions were formulated from the observation of the subjects and not from the performance data. Accordingly, none of this discussion should be assumed to be "correct" or validated by testing, it is presented for the reader to use as a basis for further research or development.

The intent was to try to generalize the observations from this experiment to the Macintosh environment. If a GUI was similar to the Macintosh, then it could be possible that the topics discussed would be relevant across platforms. The

---

[8] All references to Apple or Apple guidelines in this section refer to Apple's Human Interface Guidelines :The Apple Desktop Interface (1987), any other reference is noted.

[9] For only fifteen dollars designers' can buy the book and gain a feeling for how user interfaces are to be designed for the Macintosh. So, by giving designers more guidelines which are easy to implement, designers will be free to spend time on less mundane tasks (like being creative and adding functionality).

figures used in this discussion are not complete, and only used to assist the reader in understanding the problem. Other issues outside of the discussion have a bearing on the usefulness of these ideas and should be considered when reviewing these discussions.

5.4.1 Visual Reference

If a frame[10], around the main work area, is used in all main screens of an interface to give the user a visual reference, then a user might use the position of the frame to help position the cursor when waiting for the next screen. Figure 14 gives an example of how subjects reacted to returning to the main screen. In the Personal Organizer subjects would always return to the main screen to complete the task and begin the next task. This was accomplished by clicking on the "DONE!" button in the lower left corner of the main screen. Subjects would begin to anticipate the presentation of the main screen and position the cursor to be on the buttons' location. A subject would be fooled by the inner border found in Screen B and would position their cursor at x (in B). When the main screen appeared they would realize that that was not the correct border to have positioned the cursor and would then have to move from x (in C) to the target. The actual window size or location never changed between any of the main windows as referenced by the outside borders in A, B, and C.

Visual frames between functions should be consistent when successive actions yield positive responses. Frames should be distinct when subjects need to be aware of a new setting that is not normally accessed by the user, or is used to activate catastrophic actions (quitting without saving, shutting down the machine). It screens were consistent in the wrong context then users might accidentally perform an operation from memory and find that it performs a completely different task when taken out of its original context. Users of the Macintosh experience this to some degree when they run a macro in a program and watch it do something totally unexpected because something the macro

---

[10] A frame, like its metaphorical counterpart the wood picture frame, is a virtual demarcation that limits the focus of the users attention to what is contained inside the edge of the document window.

does not understand got in the way of completing the macro successfully. **((Add example from literature about the commands that user types that were meant to be text and wound up deleting the file and quitting the program))**

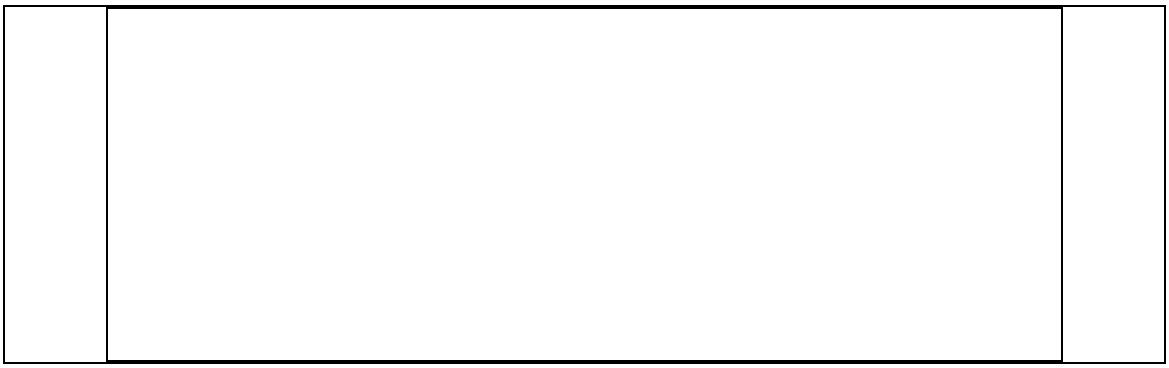


Figure 14. Visual reference- Using a frame.

5.4.2 Hot Objects

When clicking or double clicking on the object will initiate a command sequence (change to a new mode, bring up a dialog box, etc…) the object is called “hot”. If an object in a “group” of objects is hot then other objects in that group also should be hot. Otherwise those objects should not be grouped. If subjects see objects as in a group, then they expect them to act, in some form, in a similar fashion. For example clicking the ruler, tabs, paragraph markers in Word 4.0 brings up the appropriate dialog boxes. Subjects who clicked on the Location text in the area code function of the Personal Organizer were presented a dialog box. Subjects’ were frustrated when they clicked on similar objects in the group, such as the Country Code, Dial Preamble, or Time Zone and nothing happened. There should be a method for separating the hot from “cold” objects. Groups are usually created by a similar graphic design or by using similar virtual devices (e.g., a row of check boxes).

A possible solution is to highlight hot objects when the cursor passes over. This would relieve the designer from having to add additional clutter to the screen layout, a current problem in hypertext systems (Reference). A similar focus was addressed by Apples' guidelines for pop-up menus, but it did not address pop-up objects or other hot objects. Hot fields could be made visually distinct from the cold fields using different layouts and graphics. Apple suggested using an arrow to designate a pop-up menu, but no standard was mentioned for other hot objects. Since any object could be hot then one could look into the functionality of making all screen objects hot. Any object that is not ornamental has some purpose in the design, and would have more than one function or system state, as in Word 4.0's ruler. Hot objects would give users a "hidden" power that is bad in the sense that it is not obviously available, but can be easily learned[11]. With the complexity software increasing, it is becoming more difficult to make every option available to users in all circumstances. Consistency from the users' point of view might be; "designate hot objects by highlighting them when the user interactively looks[12] at the object". In reference to making all object "hot", if users understood that all objects were hot then there would be no reason to have for highlighting. If the screen had text that was linked via hypertext, this would present another problem.

5.4.3 Menus and Quick Key Usage in Dialog Boxes

---

[11] Hidden features minimize screen clutter, and allows for a wider variety of functionality enhancing the visual appearance of the interface.

[12] A subject looks at object A with their eyes, but interactively looks when the cursor is positioned over object A and looks at the object with their eyes. This idea was developed from watching many users take the mouse and make circles on the screen with the cursor, either as a sign of impatience, when waiting for a system response, or when scanning the interface for a specific object. Although an eye tracker was not used, it appeared that the subject were using the cursor as a pointing device for their eyes. This type of use was very distinct from discrete mouse/cursor motions used to initiate a specific action.

It was not obvious to the subjects that menus and quick keys were available in dialog boxes. This probably stems from the inconsistencies found in Macintosh applications. Some applications have modal dialog boxes that allow for menus and keys to be used. Others have true modal dialog boxes that do not allow actions' outside the dialog box. For example, Apple's Disk First Aid application looks like a modal dialog box, but you can use the menus. If users are going to understand the attributes of specific window type then the window types need to be used consistently.

Menu usage should not be allowed in modal dialog boxes (or they should not be called modal), but there is no reason why edit commands cannot be supported. How to inform users that edit keys are supported is the problem. Menus are clearly "outside" of a Modal Dialog box, while the keyboard (used "inside" of the box) is supported. Edit keys are keystrokes and should be supported via the clipboard. The Macintosh system software does not support the edit commands in some dialog boxes, while some programs do allow them. If all applications and the system software supported the edit keys in dialog boxes, with the system software, then this would become a simple learned response and users would not have to be "informed" of their potential operability.

The modal "non modal" box should not be used. The graphic style of a window gives important clues to their use. Having modal dialog boxes, that act like floating palettes or non-modal dialog boxes, can confuse users' to the point where they will not know what actions are appropriate for a type of window. The Apple guidelines are clear on how windows should be used, and when designers violate these rules/guidelines then users suffer. Apple says "…you should use this type of dialog box sparingly. In particular, the user can't choose a menu item while a modal dialog box is up…the main use of the modal dialog box is when it's important for the user to complete an operation before doing anything else" (Apple, 1987, p. 59). Figure 15 gives some representative samples of the most common types of windows used in the Macintosh environment (See also Topic 7: Prompt Box Locations).

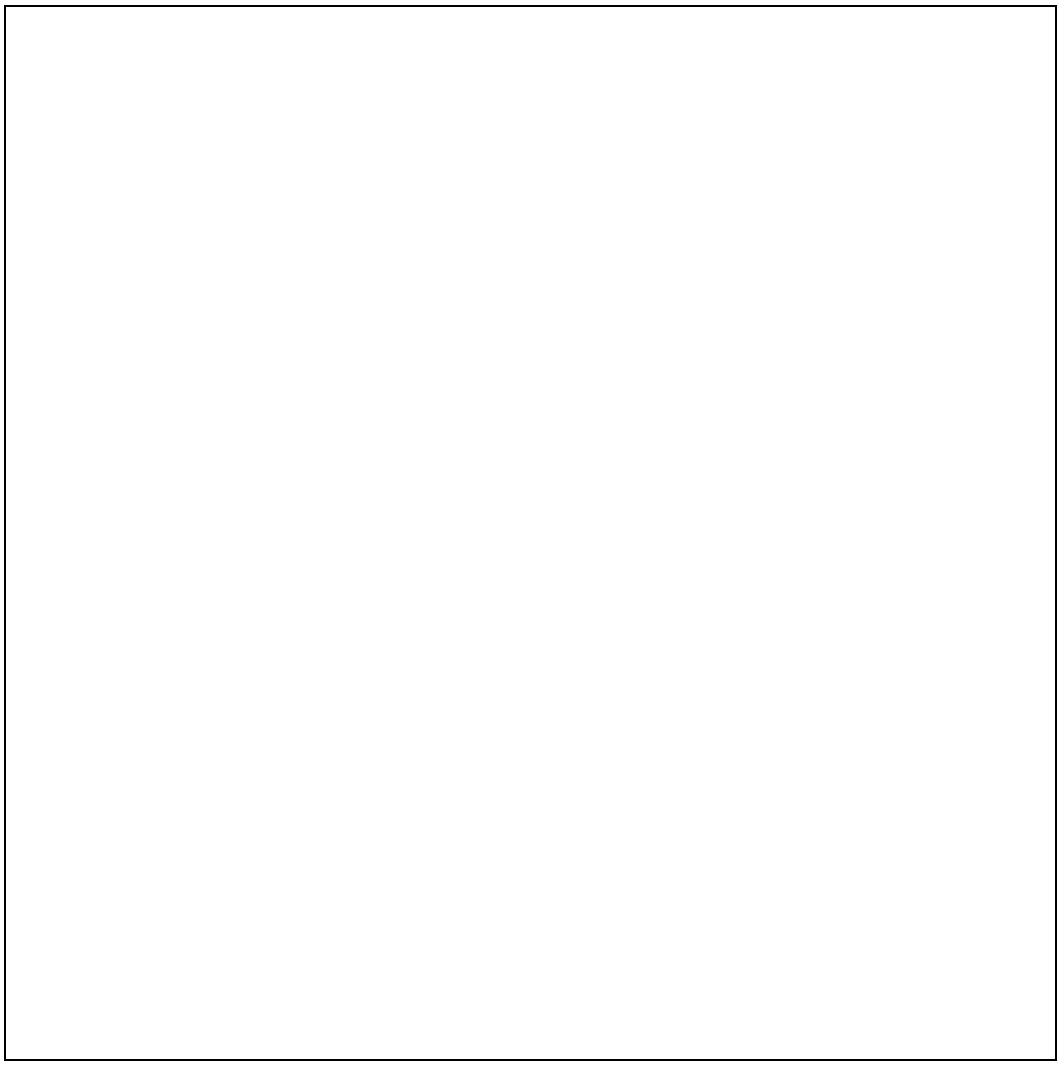


Figure 15. Window types and their usage.

5.4.4 Icon Pop-up Menus

If icons contain pop-up menus they should be identified (See #3). Potentially the small down arrow could be used, in the corner of the icon, to signal the availability of a pop-up menu (Figure 16). Part A shows the standard icon, part B shows it with a pop-down menu and part C shows the icon with the menu to its right. The use of the arrow would be consistent with the rules for textual pop-up

menus. Users' in Group 3 did not express a great appreciation for the pop-up icon used in the interface. The negative bias can probably be attributed to the slow response of the menu and not to its functionality. Pop-up iconic menus, with "universally understood" icons could have important implication for interfaces intended for an international audience. Some graphic programs use text menus with iconic choices (e.g., Cricket Graph 1.3, part D of Figure 16). One application uses iconic pop-up menus similar to the one used in inconsistent version b (they are used in the graphics layer of Nisus 3.0, part E of Figure 16). Of course, if universally understood icons were available there would be less need to translate an interface. As companies continue to expand to countries all over the world, and as networking becomes more commonplace, it will not be unusual for a company to use the same piece of software all over the world. Groupware, used by employees of different nationalities, would be a good candidate for the application of globally consistent rules. Thus a person could go from one office to another and be equally familiar with the application icons without having to know the language. Specifically the pop-up nature of the iconic menu would be similar to what is defined for standard menus.

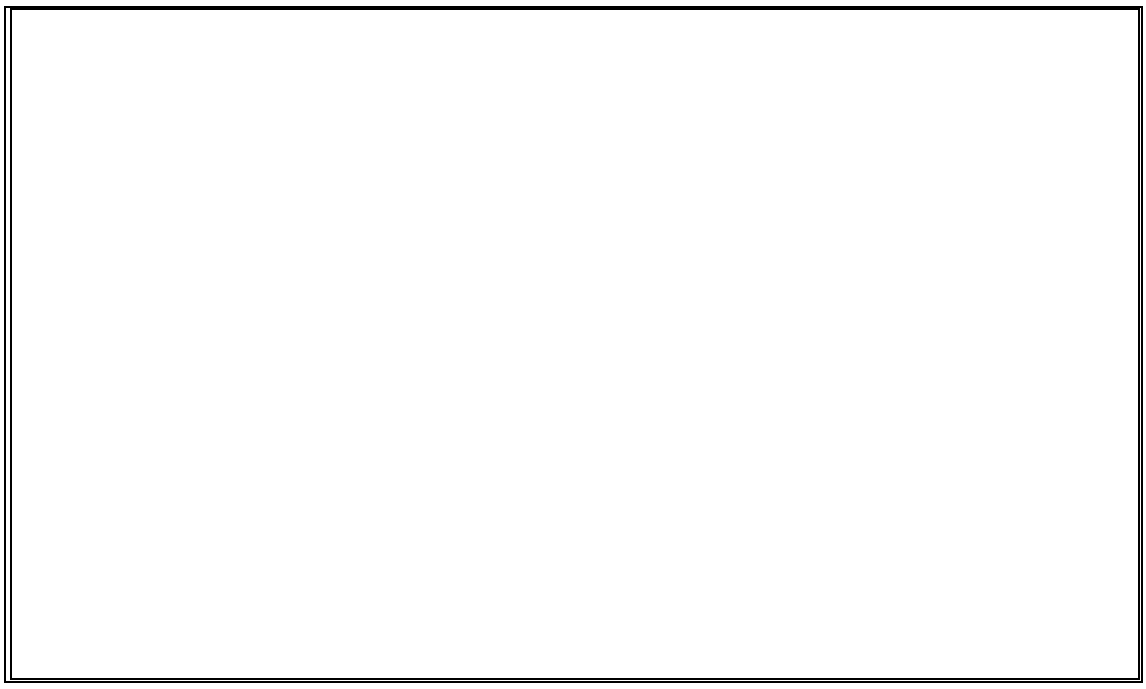


Figure 16. Example of iconic pop-up menus.

5.4.5 Icon-Menu Similarity

Subjects had objects available in the iconic menu bar (Figure 19) that were not available in the standard menu bar. Apple guidelines do support the use of iconic standard menus, but they have been rarely used in applications[13]. The function icons were placed on the screen from left to right, but a standard menu bar would have these listed from top to bottom. Users' suggested that menu equivalents should have been provided. This would make it consistent with other Macintosh programs (but not HyperCard programs that, until recently, did not support an easy method for creating menus).

The navigation arrows would not from the additional feedback provided by the screen layout of the keys. A top down menu would not give the user the same logical arrangement found by ordering the arrow keys horizontally (Figure 19).

5.4.6 Closure/Cleaning Up On Exit Functions

Subjects tended to "tidy-up" a function before exiting. In the consistent version all Help screens were automatically closed upon exiting a function. Most subjects never realized this feature because they always closed the Help fields manually before going to the next function. The Macintosh system software contains a few features that help the user clean up screen clutter such as; the zoom box, the Clean Up Window/Selection menu choice, and Command Shift-W to close all open windows. Accordingly, at the application level, housekeeping utilities should not be ignored. Operations like sorting roledex cards, closing help screens, the tile window command, and updating lists would be some examples. One needs to consider the complexity of the manipulating large files or

---

[13] A good example of a iconic standard menu in in StuffIt Deluxe from Aladdin Systems. A preference can also set the menu to appear in only text or as a floating palette in addition to the menu.

databases when determining the need for specific housekeeping operations. Users tend to push programs to do things that they were not intended to do. Sorting by first or last name, date, or company name might not be an issue if the user only has ten or 20 records, but when hundreds or thousands of cards are available then some operations turn from being convenient to necessary.

5.4.7 Clipboard Usage

Subjects were asked to move a piece of text from one week in the Weekly Planner to the next week, and asked to place a new meeting in the original week. Of course, the order in which the instructions were presented is important, but even when subjects knew what was necessary to complete the task they would copy the text, move to the next week, paste the text, move back to the previous week, and insert the text (5 steps). As opposed to copying the text, (while the text is still selected) insert the new text, and then go to the next page and paste the old text (3 steps). Subjects seem to have a set procedure and were unwilling to have to remember to move and paste after typing text. From observation, subjects seemed to have more pauses when trying the “faster” method, and subjects were unwilling to do other tasks while something was held by the clipboard for immediate use. Subjects could be “scared” of losing the contents of the clipboard. If they go and do other tasks they must remember to complete the unfinished cut and paste before accidently erasing the clipboard with something new.

The show clipboard option could be implemented, but it takes up valuable screen space and would require the user to make additional steps to complete tasks. A better solution might be to use the message bar to display the first few words of text being held in the clipboard. The Personal Organizer does not support graphics, so the clipboard message could be; “Clipboard: Jim Stevens 232 Worth Ave…”. This could be displayed while the clipboard is in use, and until another action requires the Clipboard.

5.4.8 Prompt Boxes Location

Prompt dialog boxes should be centered in the subjects’ working area, not in the center of the screen area. Due to the proliferation of large screen displays,

users have many times the screen space of compact Macintosh users (a 9 inch diagonal screen). Users' have to traverse up to about four times the screen area of a compact Macintosh user to click on prompt boxes. The application should keep track of the current location of the main window and compute an appropriate location for the prompt box. This rule might be more in line with user expectations and would clearly lessen mouse travel time. Figure 18 outlines how a user centered approach to placing prompt boxes would help. The top view shows that the prompt box is close to where the users' cursor was positioned (the large black box represents the cursor and the black line represents the distance a users' mouse has to travel to the middle of the dialog box). In the middle view the subject is now working in the bottom document and the subject has to move the cursor a longer distance to the dialog box. So, as in the bottom view, with a user centered approach, the user only has to move a very short distance to the dialog box. The box cannot be exactly over the users work space or inadvertent activations could occur, but since Apple does not document their rule for the placement of the prompt box, an investigation of the effect of centering dialog boxes in the users' immediate work space is needed. This makes the placement of the dialog boxes consistent with a new rule for their placement.

This type of user-centered rule is problematic and might not be appropriate, but many applications are not consistent with their use of modal dialog boxes. Many programs center the simple dialog boxes like in Figure 18-A, but left justify the complex Save As… dialog box while others center the Save As… dialog box. The rule structure is unclear. The user-centered approach is not without it's problems. First, the subject would always have a different location for the dialog boxes depending on the current location of the document window, and second, the dialog box might interfere with viewing the document. Obstruction of the document can cause the user to have to cancel the current operation because the dialog box is in the way. This occurred in the inconsistent versions of the interface when some dialog boxes were positioned to interfere with information needed to determine the correct response to the dialog box. This also happens in some word processors when Spell Check windows or Find dialog boxes get in the way of viewing the screen.

One subject suggested that modal dialog boxes should have been movable. A title bar could be integrated into a new type of dialog window, which could be moved but would still prevent a user from continuing until it was dismissed. Recently a new Apple guideline outlined the use of a movable modal dialog box, which is a combination of the visual features of a dialog and document window. Figure 17 shows an example of this new window type. Note the combination of the title bar from a document window and the thick black border from a dialog box.

Non modal boxes and moving modal dialog boxes should remember their position as outlined by the Apple guidelines. This allows a subject to move the position of the dialog boxes to be more convenient, but it will decrease a subjects ability to remember the position of the box and to click and type ahead. Since these types of boxes are usually much more complex than the standard “Are you sure you want to delete this card?” prompt box, more time is spent by the user in making selections and typing information in the box, so the ability to type and click is not as useful.

All three of these discussions; a user centered positioning rule, a moving modal dialog box and having these boxes remember their previous positions could help to speed user performance be lessening mouse travel time. In a recent upgrade to ResEdit [14] a new option allows developers to position dialog boxes according to their choice of rules. Although no guidelines are available at this time to use this feature, it will probably be available in the near future. The Dialog “Auto Position…” feature allows the operation system to automatically position the window by four rules; None (no rule), Center, Alert Position, and Stagger. The last three choices can apply to the Main Screen, the Parent Window (the document window in these discussions), or the Parent Window’s Screen. The allows for dialog boxes to be positioned according to the rules discussed, but it does not support the rationale. Even with the ability to

---

[14] ResEdit 2.1 allows developers and advanced Macintosh users to manipulate and create the objects used by the Macintosh toolbox. The references to System 7.0, Apple upcoming revision to the Macintosh operating system, are only speculative. This could change before the software is officially released.

implement these ideas, Apple has not released any information on how appropriate these ideas would be if implemented.

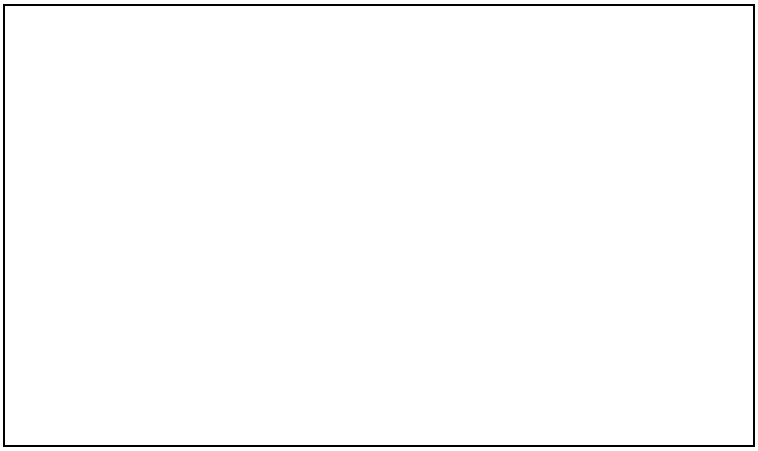


Figure 17. Movable modal dialog box (Jensen, 1990).

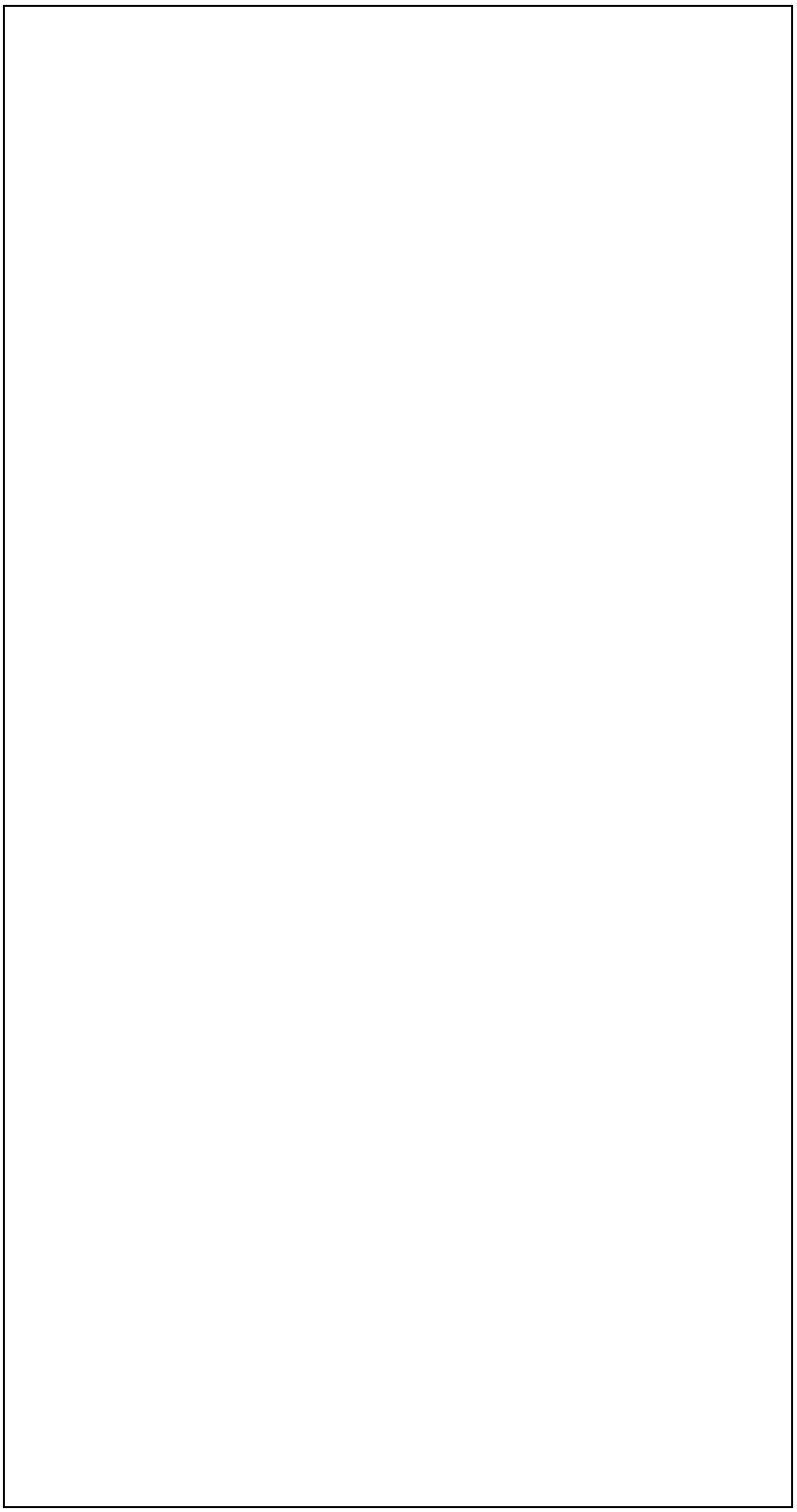


Figure 18. Positioning the prompt box from a users perspective.

5.4.9 Return Icon

The Return icon, used frequently in hypermedia programs, is not a linear or multiple-linear motion device like the previous and next icons (linear), fast forward and fast rewind (a multiplier * a number of linear jumps), or first and last icons. The Return key, unlike the other features, is not intuitive in its function. Users are constantly exposed to the other iconic representation when using VCR control panels, cassette players, and other everyday items. All subjects' used the button in one of two ways. When subjects were unfamiliar with it's function from previous experience they created a rule from the way it functioned when activated. Unfortunately, some subjects used it once, generated what they thought to be the correct rule, and then found out when they used it again that the rule was wrong. Subjects mentioned that the return icon would take them back to the main screen, as opposed to its actual function to take the user to the previous location in the previous function. When users went from the main screen to any other screen and then used the Return button, both rules were applicable, but when subjects started to go from function to function, without returning to the main screen, they then would realize its true function. Thus, subjects who used the Return key after just leaving the main screen, reached an erroneous conclusion about the Return keys function.

The Return key is a hypertext function and should be distinctly separated from the previously mentioned functions to distinguish it from the other navigation icons. Although a return key was available in the consistent version, and its use would improve timed performance, very few subjects used this key. Some subjects even commented that there should be a way for them to move back to previous locations (that was the function of the return key). Figure 19 shows that the return key was not separated from the other arrow keys in the consistent version of the Personal Organizer.[15] Other hypermedia applications have the

[15] The "previous card" and "first card" icons are greyed out, and unavailable to reflect that there are no cards prior to the current card.

return key in a lower corner of the screen, away from the other navigation icons to set it apart from the other navigation keys. As with other icons used in the interface, the Return icon, being similar to the arrow keys, but distinct, should be separated from the other arrow keys without totally separating from the other arrow keys. It might just need a small line between it and the other arrows keys to give it special recognition.

Figure 19. The return icon in the consistent versions iconic menu bar.

5.4.10 Editable Looking Fields

Often editable fields are placed near non-editable fields. For example, in the area code function (Figure 20), appears to be editable, but three types of fields present, only one needs to be edited. The **Bold** words represent titles for the fields, and are not editable. In most circumstances, additional locations would need to be added frequently, as all locations are not included in the list (like smaller communities, lakes, or “Jim’s Beach House”). These can be called a fully editable fields and would support all edit operations without and interference form the application. Area codes rarely change and the other information probably would never change. This can be called an infrequently edited field, and might require some form of prompt to keep the user from making serious mistakes with

this field, while still allowing the user to copy the information in the field. In this case, clicking on **Region** brings up a dialog box for quickly jumping to other regions, but clicking on “California” or “408” will allows the user to edit the field. Some subjects assumed that by typing the name of a region or area code into the corresponding field it would make the program jump to the specified card. In fact, this changes the current field’s name or area code without prompting the user that they just changed the current cards region or area code. So, if the subject typed “Alabama” into the region field and expected the software to move to the Alabama card, it did not, but it would change the name of the California card to Alabama. When the subject revisited this function and searched for Alabama, this card would produce a hit, although it would clearly contain incorrect information. Some form of a lockout feature would be needed to protect users from entering spurious information. A complete lockout is not functional since users need to be allowed to copy text from the fields to avoid having to retype the information elsewhere when it is needed. It would then be consistent to have a lockout feature for the Locations field, but since this field would be edited frequently it probably would be inconvenient to ask for permission every time a change is entered. Guidelines for preventing user errors were documented, but guidelines for making distinctions for fields (beyond pop-up fields) are not available. Grouping like objects, Bold text, and possibly the ubiquitous “…” used in menus could all be useful for making distinctions between the types of fields. The titles would all be hot and would invoke navigation functions similar to the one used when clicking on “**Region**”. As a function the overriding objective of the system would be to not allow the user to make catastrophic errors without prompting the user to the seriousness of their actions. Figure 21 shows what a possible prompt might look like if a subject tried to change the information found in an infrequently editable field. This would allow the user to use the field to navigate to another page without changing the name for the region. Numerous users typed in the Region field expecting to have the program take them to the appropriate page.

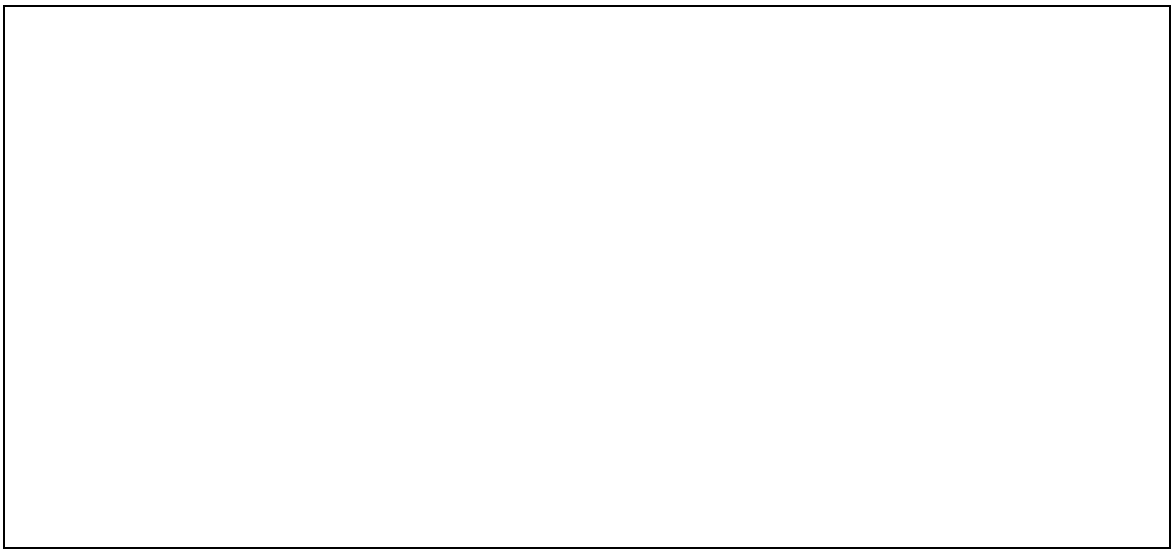


Figure 20. Three types of fields (in the Area Code Finder).

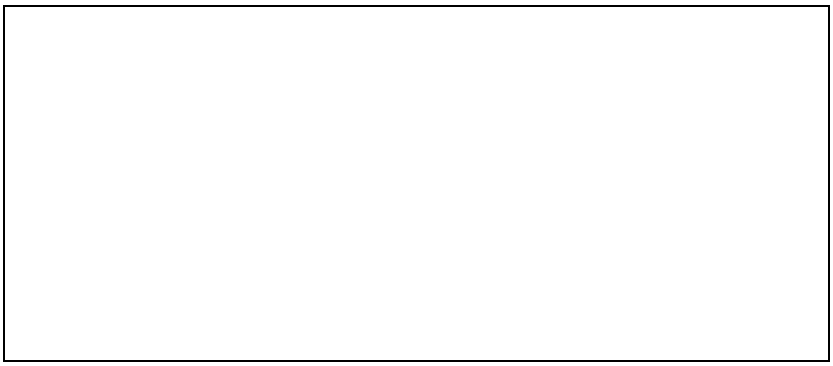


Figure 21. Prompt for determining a users next action in an undesirably editable field.

5.4.11 Help Length and Breadth

Help systems and many other topics are well documented in the literature. Some of this has been translated into guidelines and rules for creating usable help systems for the Macintosh. Using the rule that help should be *Short and Concise or Long and Precise*, and allowing the option to switch between the two levels could be a start. Concise implying descriptive using technical terms that a Macintosh user would be accustomed (click, double-click, Find, Etc.), while precise would imply language without "Mac-speak", to assist those who do not understand the concise information. The complexity of help systems are another area of concern when trying to accommodate expert and novice users. The short and long distinction might be one way to help solve this problem. The ability to expand helps detail should be available from the help window (Figure 22). It seems more useful to expand or contract help for individual pieces of information, than to have a system mode that makes a user stay in one mode until a change in a preference value. The option to switch should always be available from within the help information window. The current state of each help entry should be retained until the end of the session. A setting could allow a user to continue with the old system states when returning to the application.

The help system in the Personal Organizer used small (1 to 8 inch square) boxes to contain brief descriptions of an object's function. Almost all objects had some help available, of course that was an inconsistency (e.g., almost all and not all).

Another inconsistency was the help for menus. It appeared in a movable window, and help for icons and objects appeared in the main window, sometimes overlapping parts of the screen. When a user would move from screen to screen within a function, or would go to a different function the help boxes would disappear (but not the help window for the menus). Only a few subjects ever used the help for the menus and they usually found it by accident.

A find function, within the Help system, also should be supported. Apple created a draft for help system principles, guidelines and some structural and context rules (Apple, 1990), but needs more detail to assist designers in understanding style or functionality issues. The bulk of the draft reviews previously defined general system features, i.e., on-line help should be see-and-

point (instead of remember-and-type), that direct manipulation should be supported, and that help should be context sensitive.

By moving all the help information to one window, and by supporting option click help on objects, consistency with user expectations could be upheld. Option-click help systems limit the number of keyboard equivalents that can be used for complicated programs. Although one might argue (and convincingly) that any program that has quick keys for function keys, command keys, shift-function keys and shift-command keys has more than a user could remember. A better option might be to incorporate a status bar, where a help mode would give context sensitive help automatically when the cursor is passed over objects. This is discussed in more detail in the next topic.

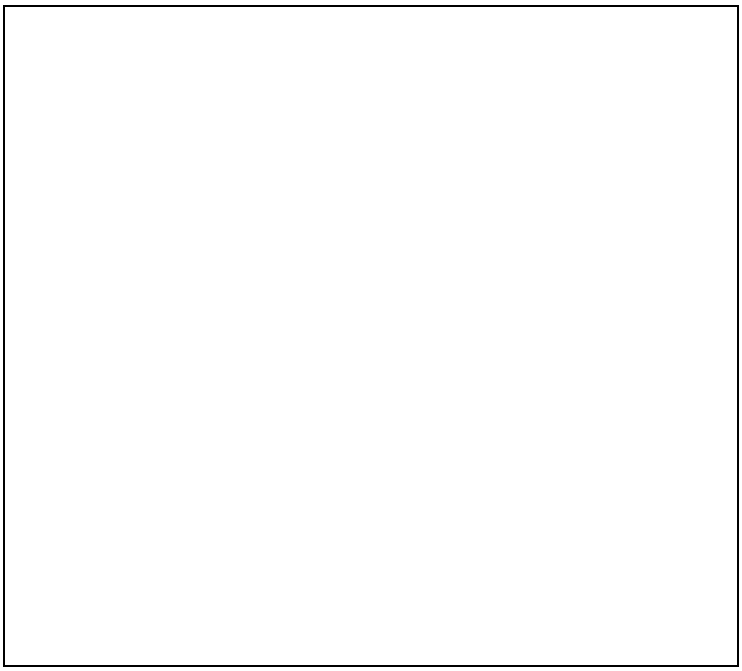


Figure 22. A possible help system for the Personal Organizer.

5.4.12 Using Help

It probably comes from the adage “If it doesn’t work then read the manual”, but subjects were hesitant to use the help system. It could be that the help system was unsatisfactory for the users, but subjects expressed satisfaction with the simplicity of the consistent versions help system. Subjects were “stubborn” concerning using help to assist them in completing a task. Subjects were instructed to “complete each and every task in a timely manner”, so perseverance was dictated by the instructions. Subjects were told that using the help system was faster than randomly experimenting with the interface. In addition a small sign above the bezel of the monitor said “Use Help it Helps!”. So, users were encouraged frequently to use help, but they generally waited until all options were exhausted before getting help. This leads to the idea for a non-intrusive “introductory” level for the application. This would be an option that could set the system to give context sensitive help, immediately, in the message window. Short help messages would be presented as the cursor passed over the location of icons, menus, or fields. A help window also could be displayed in this mode (like Figure 22), so users would realize that more extensive help was available. Additionally, function names could be displayed under each icon in this mode. This would clutter the screen, but would only be used by the subject until they were accustomed to the interface.

5.4.13 Distinction of Fields for Navigation

In a database program the tab and shift-tab keys are used to navigate from cell to cell. It was not clear to the subject what fields were available for navigation with the tab key in the Area Code function (Figure 20). Using the tab key adds to the number of keystrokes needed to complete a task, but it can decrease the amount of time needed to complete the task. Since the Tab key presses, like all keyboard inputs, are held in memory and can be cued by the computer (type ahead) for subjects who are familiar with the interface.

So, one change could be to arrange the fields to follow a smooth pattern when using the tab keys. Tab key movement should occur from left to right and then top to bottom (as defined by Apple), and the organization of the fields should

facilitate understanding the logic. Editable fields should be grouped in some order, while non-editable fields should not be in the middle of rows or columns that are accessible with the tab key. A logical arrangement that should be recognized when the tab key is used. A rule should define why some fields were editable and others were not. This rule should become apparent to users. (i.e., Bold titles with "…" can be clicked to bring up dialog boxes, and mousing over a data field turns the pointer cursor to a text insertion bar). This type of rule might help define the grouping of hot fields. Both rules would help to foster a sense of consistency in navigation between fields and a consistency in the "hidden" functionality of the data or title fields.

5.4.14 Adaptive Interfaces

A more adaptive and intelligent application could assist the subject without the addition of complicated code or artificial intelligence. Search, inter and intra function navigation, and other features could have incorporated adaptive technology to make the user more productive. This productivity increase would be a function of allowing the system to increase the subjects' knowledge in understanding the interface, assist the subject in making proper choices when searching, correct typographic errors (in some cases), and to help the user understand the best navigation strategies.

The more user testing that is done by developers the better their understanding of the types of errors that could be avoided or at least minimized. For example, subjects sometimes used the arrow keys to navigate between screens in a function. When the number of screens that needs to be scanned was small this was an effective method, but when a subject had to move 5, 10 or 20 screens then another method of navigation would be more effective. A numeric counter, in the application, could be set up to detect a specified number of repeated clicks on an arrow key. If the counter reached that value the computer would prompt the user that a faster method of navigation was available. The computer also could give a demonstration of the more effective navigation strategy. Research would have to determine the appropriate number of successive clicks and the time between each click needed to activate such a help feature. If the subject typed in a field when moving between pages the

counter would reset. If the subject moved quickly (rapid clicks on the arrow key) it would trigger the computer to show the user who to use the quick week jumper operation. This is under the premise that the user is moving quickly to go to a specific week, since the user is clicking “faster” than he or she could scan each page for a certain piece of information. This speed would have to be calculated from an experiment. If the subject was moving slow, then the user might be scanning for a piece of information, instead of looking for a date, so the computer could prompt the user on how to use the “Find” function to locate a piece of text. Expert users should have the option to turn these features off, since they would know how to use the most efficient form of navigation and might have different needs would would be hindered by this type of interruption.

5.4.15 Breadth of Consistency

The more consistent the interface, the more users will expect from the interface. Designers must approach the rules to building interfaces from the users point of view; “If I can do it in X why can’t I do it while using Y”. The more global the rule, the easier it is for the user to make assumptions. If a user navigates from function to function and has to learn new “consistent” rules in each function then the interface will not be consistent between functions. The application and not the function level might be a more important level to apply consistency if the user will spend an equal amount of time in each function. If only one function is frequently used then the function level consistency might take priority. This needs to be defined for the entire population, not just a specific group of users. Rules will need to be applied globally, or users will be less inclined to user a variety of software. Emphasis on the wrong level of consistency can have pronounced effects on the commercial viability of the software. The Word Perfect Corporation tried to enforce consistency across its line of word processors but creating a “consistent” version of the software for the Macintosh environment. It was consistent with the other versions of their word processors, but was radically different from Macintosh applications. Users rejected the software, and they have spent years redesigning the software to reattempt entering the Macintosh market. So, the level where designer’s apply

consistency becomes as important as the logic than goes into defining the interface.

In the Personal Organizer, the subjects had the Find feature available to them in the Weekly Planner, Address Rolodex, To Do lists, and Area Code Finder. It was not available in the Phone Setup function (with only one screen there was nothing to search), the Six-Month Calendar (a search would have only been available for the year, a day of a month, or the abbreviation for the month, but not for combinations of any of these, thus making a Find function essentially useless), and in the Dial Number window (an oversight by the designer). Many subjects commented on wanting a Find feature for the dial number window, but no one expressed a need for one with the other two functions. Subjects realized the need in the later case, and the worthlessness in the former two cases. Adding a Find option for the Help system (and the Phone Dial window) would be consistent with what users requested without having to add the Find option to the Phone Setup and Calendar.

5.4.16 Locus of Control/Font Selection

Users were not very aware of when and how font, style and size (FSS) changes took place. Most users probably experience some problems with either highlighting text and choosing a FSS, or placing a cursor on a line, not knowing until they start typing what the FSS was for the text. In this application it was possible to have a cursor insertion bar not visible until the subject clicked inside a field. This caused many users to assume that FSS changes would affect the field when they entered it after selecting the FSS. Since fields already had an existing FSS, this caused problems for subjects in determining the appropriate settings. All programs make the user set a FSS change while in the field for it to take effect. This is a less than optimal solution since subjects might not be aware of what typestyle would appear , and unless one wanted to check each FSS menu it would be a hit and miss situation. A small TYPE window could be used to report the current FSS information, and that information could be represented as the current choices on pop-up menus. This is not consistent with Apple's Font, Style and Size menu, but since every developer sets these up

differently there is no reason not to use a floating palette, if it was more comfortable for the user. For a user with a large screen, or for a user who makes many FSS changes, it could reduce time needed for viewing the FSS menus. The standard quick keys also could be supported. A quick key (to open the font menu and hold it on the screen) could be used with the arrow keys to select a specific font without the users hands leaving the keyboard. Microsoft Word 4.0 offers this feature for their menus[16], but this would be available for the palette. Figure 23 gives an example of what a Font Manager could look like (in A and then active in B) for changing or just identifying the current selections. This would be a non modal dialog box, so it could always be visible. The example of the Font Status window (C) could be used if screen space was at a premium (a smaller fonts could be used). This would keep users from making frequent trips to the menus just to check on the FSS status. Multiple styles could be noted by a "Multiple" notation or by having multiple styles listed in the window. Another solution involves using a portion of the horizontal scroll bar to display a sample text string styled like the currently selected FSS (Figure 23, part D).

[16] Holding down the . (period) keys on the numeric keypad (with the number lock key off) and then pressing the first letter of the menu, or by using the arrow keys, will display the menus. By using the arrow keys or by pressing the first letter of the menu item name the selection will be highlighted. Pressing Return will activate that item. The Escape key will cancel the menu selection process.

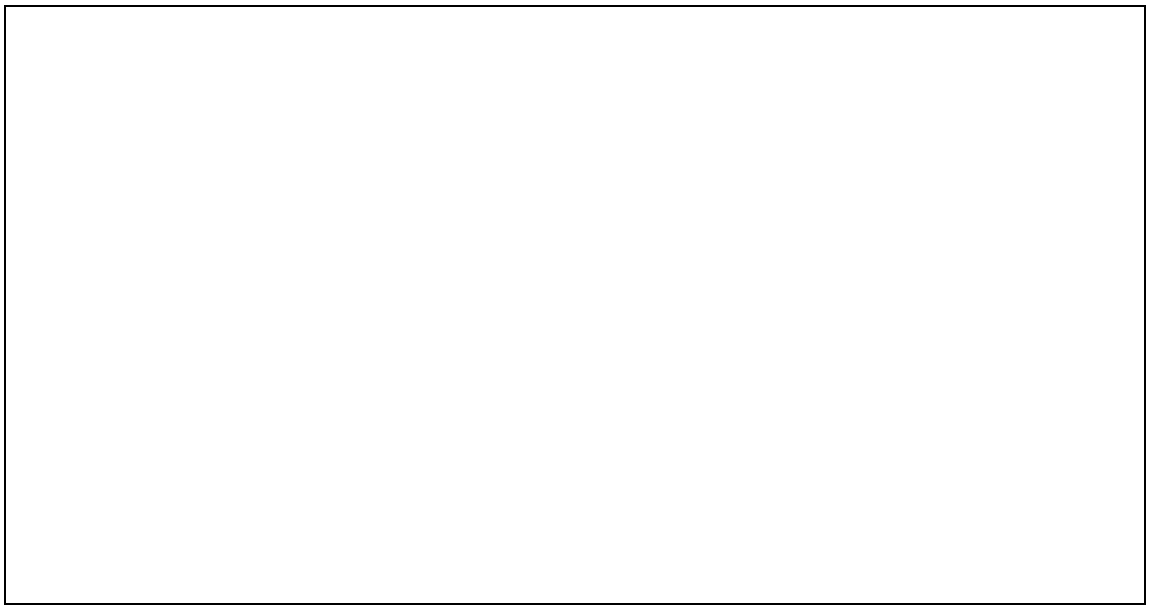


Figure 23. A font manager, font status window, and font status line.

5.4.17 Screen Consistency (Similarity)

Occasionally a subject did not see that they were entering data into a Weekly Planner page in the wrong year. Although the year is shown in the top left corner of each planner page, the subject did not realize that they had gone too far ahead to find “August 21”. Since users would work in a single year for most of the time (except in the last few months of the year when users might start scheduling for January of the next year) some form of additional feedback should be used to tell users that the planner pages look the same, but were from different years. It would be easy to make the Year blink two or three times when a subject moves from a page in one year to a page in another year. This additional feedback might be applied to other circumstances where consistency in the interface design makes different screens look nearly identical, thus giving the user some additional way to differentiate the screens. This type of “intelligence” could help prevent user errors (Also see Adaptive Interfaces, Topic 14).

5.4.18 Generic vs. Global Messages

The Macintosh is well known for giving descriptive messages, but it makes it even more annoying when it does not. “Delete the visible address card?” could become “Delete the visible address card for ‘Jerry Garcia’?”. It takes little extra effort to take existing information from the application and incorporate it into more useful messages.

5.4.19 Acceptable Navigation Methods

If users have problems navigating to an appropriate position (either by arrow keys, find strings, or pop-up lists) then it might be due to the designer’s implementation of the search strategy. In the Personal Organizer, subjects sometimes found it difficult to find the correct card for an area code, state, or city. The methods used to complete the searches varied depending on how the user approached the problem. Specific navigation methods were inappropriate for

some searches and other methods, that were not implemented, might have been more useful. Table 40 lists some of possible search methods.

It was not clear what circumstances make it more appropriate for one method to be used over another. One can look to the research literature for some answers, especially concerning hypertext, but when the research has to be applied by developers it needs to be stated in an easily addressable and verifiable language. For example, when should a scrolling list be used in place of pop-up menu for making single selections? How long would the list have to be, to make one choice unacceptable? Which is faster? The research generally focused on breadth vs. depth issues for menu hierarchies, and not on the appropriate form for navigation.

TABLE 40. Potential Methods for Navigation.

Forward and reverse (single step)*	Find (search string)*
First and last*	Find Again*
Fast forward and fast previous*	Hypertext links
Return to previous location*	Pop-up menus
Recent pictorial history (HyperCard)	Pop-up icons
Bookmarks (a form of Hypertext)	Scrolling lists
Jump to (page number)*	Radio or Check boxes?

* A navigation method that could use a keyboard equivalent, menu choice, or a mouse click on an iconic for initiation.

5.4.20 Users "Jump" Before They "Look"

The subjects create mental models of the system, often they worked properly, but some subjects were so fast that the system could not properly respond to all

their inputs. Subjects would attempt a series of commands, and would have to wait until they were completed before realizing that they were the wrong actions to apply. At other times, while typing the commands, a subject would make a mistake and would have to wait until the mouse or keyboard entries were processed by the computer before they could correct their error. After seeing that their actions were unsuccessful in getting the appropriate response, they would try to accomplish the task properly. This is not uncommon, and stems from the localization of consistency, i.e., the inability of users to make assumption about when rule structures could be applied outside the immediate surroundings. Most Macintosh software allows for this form of experimentation without any data integrity penalty. The user might lose time, but should not lose data. Cases where this would not hold, are when consistency is placed in front of common sense. For example, users frequently click ahead on dialog boxes to complete tasks. It would be inappropriate for designers to make dialog boxes so consistent in their placement that unrecoverable actions could be orchestrated by unknowing users who, with muscle memory, were attempting to complete a completely different action.

5.4.21 Consistency Across Applications

Subjects who received the inconsistent versions of the interface moved the menu/icon palette to the right of the screen more than to the left. This held for left and right handed subjects. Since all subjects were experienced, it is not a difficult assumption that all subjects had some experience with graphics programs. Since most graphics programs use some form of a icon palette either attached to the left corner of the screen, or originally positioned in the left corner, subjects were familiar with this orientation, and should have chosen to position their palettes similarly. Subjects positioned the palette to the application right, possibly to be “closer” to the mouse (almost all users positioned it to their right), or possibly because the application was positioned close to the left edge of the

monitor and it would have been a tight fit to position the palette[17]. Thus, subjects might have felt that placing the icon palette on the left would "cramp" the screen area, even with enough room. Wanting to "get it out of the way quickly" is probably the most logical reason, but in either case, it favors allowing subjects to move the palette to a position that they find convenient. This would contradict the consistency rule of "The palette shall be positioned in the left corner of the screen, so all users will know where to find the palette", with the new, user centered rule, "The palette shall be positioned wherever the user finds convenient". This is beneficial to users with large screens, who find palettes that are locked in place (e.g., MacDraw II) to be too far away from the work space. Users' will probably try to position the palette to minimize obscuring the document. Also, the option for horizontal and vertical orientations should be available for those working in landscape or portrait modes.

Since the order of the icons in the palette is constant, the subjects will still have a visual map of the palette icons, the potential decrease in productivity due having to keep track of the position of the palette should be more than offset by the shortened mouse travel time. Too many palettes also could be of concern. Making every function visible on palettes could leave little room for the document window.

5.4.22 Feedback

Feedback plays an important role in creating a good user interface, but sometimes it gets in the way of productivity. Often, systems will respond with the dialog box "Search String Not Found" with an OK button (See Topic 24 for an example from the Personal Organizer) or even just a beep. These types of dialog boxes can be disruptive to the users' train of thought. They should be reserved for catastrophic operations and not just for the presentation of information. If a user is about to delete a file or do something that is unrecoverable then this type of interruption could be warranted. It would break

---

[17] Most programs which have a floating vertical icon bar, position the document window about 2 pixels' from the palette, accordingly there was adequate room to position the palette to the applications left.

the users train of thought, but that is the intention of the interruption. When searching for strings of text, or trying to navigate, there is no reason to stop and make a user “prove” to the system that they have read the message. A message box, as in HyperCard, can be a much more productive way of giving users feedback about non-critical events. A subject could read the feedback message without having to respond to the dialog box. A “bad” beep could accompany the message to alert the user that the message was present, as with the dialog box, but it would be much less intrusive, and it would allow users to continue working. It also would place the status of dialog boxes higher up in the “importance” hierarchy. Thus when a dialog box was presented, the message would convey more importance, and not just signal another interruption. Other programs do support message bars (Microsoft Excel, and Silicon Beach SuperCard). The option to hide the message bar should be available to minimize screen clutter when not needed by the user.

5.4.23 Find and Replace - Search Theory

When subjects searched for text in a function, sometimes their search sting was not found and the system would respond "'Betty Carr' was not found anywhere in the Weekly Planner". Although the system responded by placing the actual search string in the prompt, and used the word "anywhere", subjects did not believe that it was not found. Subjects would go to the beginning of the function (the first week in the planner) and try the search again, under the assumption that the system was not searching anything before their original location in the function.[18] Potentially the word "anywhere" could have been in all uppercase, but this brings up another point concerning the search space. Users frequently use the Find command in a variety of programs and the method for initiating a Find varies. In a word processor subjects can look either in the entire document or in a selection of text, but in Microsoft Excel, a spreadsheet application, the user must specify the search area before defining the string. One solution for this would be to create more powerful Find functions, giving users more flexibility in specifying search strings, and more confidence in knowing that a string was really not found. Some guidelines for developers to follow for creating a "Super Find" function would be very useful. Figure 24 gives an example of what a possible enhanced find function could look like for the Personal Organizer. Boolean logic also would make a good addition, although it also would increase the complexity of the design and the user's task. The Find/Replace window is non-modal for verifying search parameters and allowing

---

[18] The HyperCard metaphors relies of the continuity of the cards, so the next card after the "last" card would be the first card . The Personal Organizer had clear "First" and "Last" cards, but the author does not feel that either having a distinct beginning and ending, or having them function as a continuous stack would affect this discussion. Circularity could effect a subjects interpretation of where they were in a stack (i.e., if the previous card was 'Steve Willis' and the next address card was for 'John Abrahams', the user might question if he/she was at the beginning of the stack, or near the end of an unsorted stack of cards. Other cues can be used to assist in that determination.

for corrections and fine tuning without having to reinitiate the Find command. The message box would be used to give feedback.

Boolean searching creates more complex searches, but with the increasing complexity of document and their increasing size, it can become an important part of creating a functional and usable interface. The System 7.0 Find allows novice users easy access to the Find function, while giving users the option to expand the hidden complexity of the search function to include more detailed search parameters.

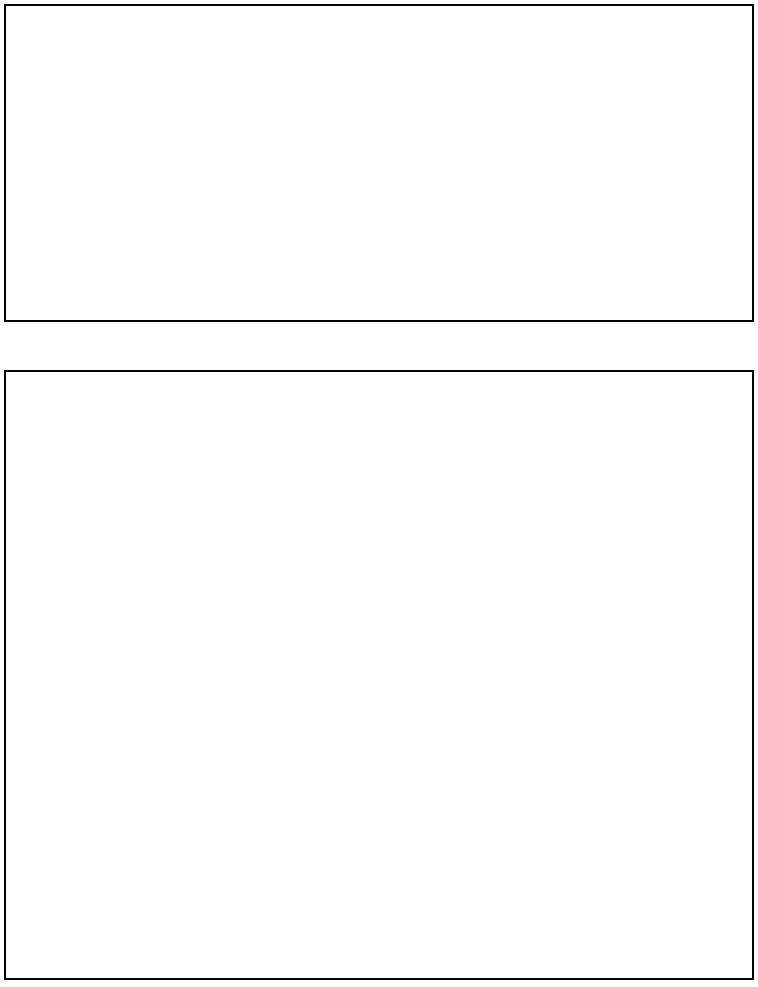


Figure 24. A potential find/replace window for the standard find function.

5.4.24 Movement by Multiples of Other Than One

Users can readily understand going to the first, last, previous, or next page when the metaphor is clear (when a page looks like a single page in a book, or a week in a year), but when the navigation involves movement other than simple motions then understanding decreases. Setting aside the problems of hypertext jumping, and looking at only linear motion, one still senses some confusion by the user. A good example was of subjects who used the arrow keys to move between pages in the To Do list. Each screen displayed two pages in the To Do list, so clicking on the "Next" arrow key essentially turned the To Do lists two pages (from 1/2 to 3/4). So, to move to from page 1/2 to page 5/6 the subject would click the arrow key twice (1/2 -> 3/4 -> 5/6). Unfortunately, most subjects overshot pages 5/6 thinking that they needed to click on the arrow key four times (1/2 -> 2/3 -> 3/4 -> 4/5 -> 5/6). Microsoft Word uses this second method to navigate with the page up and down buttons while a subject is in the "Page Preview" mode. In Word 4.0, the screen will scroll one page down when the key is pressed even if the display is in a two page side-by-side view. This means that to see the next two pages the subject has to hit the page down key twice. Consistent, yes, but consistent in a context that the user might be uncomfortable in accepting. The navigation method should be matched with the metaphor used in the function. If the subject was reading a hard copy book (with text on the front and back of a page), the user does not turn only one page at a time, but two pages at a time from the current two pages to the next two pages. The turning of pages could be presented to a computer user as an animation that mimics the flipping of pages in a book. Although, this type of animation was not used in the experiment, it is available in the HyperCard and SuperCard applications. Furthermore, software like PageMaker 4.0 uses miniature page icons, aligned across the bottom of the screen, to graphically depict what pages were currently visible, what pages were in the document, and what the page numbers were for each page. Clicking on an icon will navigate the user to the selected pages. Figure 25 shows what the icon/page bar looks like in Pagemaker 4.0. Note how the turned corners denote how pages would face each other when printed as a double sided document.

When movement using other than a jump of a single page or a jump of a distance defined by a well known metaphor (like the 7 days of week) user testing should verify that users would not make unnecessary errors. Something as simple as having facing pages could be a problem, as in Word 4.0, if not addressed. As a further note, the Page Preview mode in Nisus 3.0, a critically acclaimed word processor, moves two pages when the scroll bar is clicked in the side-by-side viewing mode. So, there is no real precedence, or rules to follow, beyond some simple guidelines presented by Apple in the HyperCard Stack Design book (Apple, 1989) when developing complex navigation strategies for the Macintosh.

Figure 25. An icon-menu bar for navigating to specific sets of pages.

5.4.25 Extended Influence of Hot Icons

Giving the user a true direct manipulation interface and not just another GUI (there is a big distinction) allows the user to explore the potential of the interface. If “every” object on a screen had some purpose beyond the apparent presentation of information (e.g., a ruler displays the current margins, but by clicking on the ruler the subject can change the margins), it could lead a user to understanding the intricacies of the interface. This extended usability would enhance, not only the functionality of the interface, but the users’ ability to teach themselves. It could create curiosity in the user, which Apple finds to be a good trait. This alone can have a profound impact on the amount and type of support and service a developer will need to provide to the users. Clearly a more intuitive interface will require less support by the developer, thus lowering their cost to

support the product. The Word Perfect corporation used to boast in advertisements that their phone bill (for customer support of the Word Perfect word processor) was in excess of $200,000 a month (and that was many years ago). Having such a large phone bill also says that many people were employed to solve problems. One can assume that a more intuitive interface, one that would generated fewer questions and be better understood, would cut support costs. So, consistency can affect the time to complete a task, and the users' subjective opinion, but it also can affect the cost to support and maintain the application.

5.5 Decision Tree for Determining the Application Usefulness

The objective and subjective measures gave conflicting signals. If subjective opinions are poor (bad first impressions) then the user might not be inclined to use the software and would deny themselves from using what could be a productive piece of software (once they adjust to the interface). A user might have a bad subjective opinion of the interface only to find out after using the application that the time it takes to complete tasks is also poor (assuming they have a way of recognizing that their time to complete tasks is actually poor and not just perceived as poor).

If the user has a good first impression of the interface (high subjective ratings) then the user will probably start to use the program and find out if the programs functionality compares to their subjective option. If the user can recognize true objective performance then the user will probably continue to the use the application, otherwise the user will switch to another application (if available). Figure 26 summarizes the possible outcomes for the user and application when considering subjective and objective measures. The subjective opinion can be developed by taking functionality into consideration, but the approach used in this research used subjective measures to establish how the user reacted to the application. The objective measures were useful in establishing the productivity of the user. In other conditions time might not be the best guide for establishing performance characteristics. Measures of errors or the number of tasks completed in a specified time period might be more appropriate. One might use

questions that relate to time measures like; “Can the interface complete a data base search in under 10 seconds” than “Do you like the way the application performs database searches?” Although this can be done it is probably better to keep the opinions and performance measures separated. Figure 14 demonstrates a flow diagram that was formulated from three possible bipolar conditions; Subjective Opinion, Perceived Objective Performance, and Actual Performance. Subjective opinions are formed first, sometimes even before one uses an application. The proliferation of computer magazines has created as plethora of “computer experts” who are paid to give their opinion of software. Usually the descriptions of performance is of little value at this point since each user can have distinct needs which are not reflected in these rankings. Except when the software is completely panned by reviewers their will usually be at least one niche that can be exploited by the vendor to show superiority for their product over the competitor. By using the software, or a demonstration copy, before passing judgement on its functionality the user can gage how well the program will work in their surroundings. In most cases one would only get subjective and perceived objective measures, but if one goes to the trouble of taking some objective measures then one might find the program more useful than originally assumed. With this in mind a more informed decision can be made about its usefulness. Although a good graphic user interface might make the program look attractive it might perform as well as another product. Conversely, many programs that are ridiculed as having poor user interfaces might provide exceptional objective performance after an initial inspection.

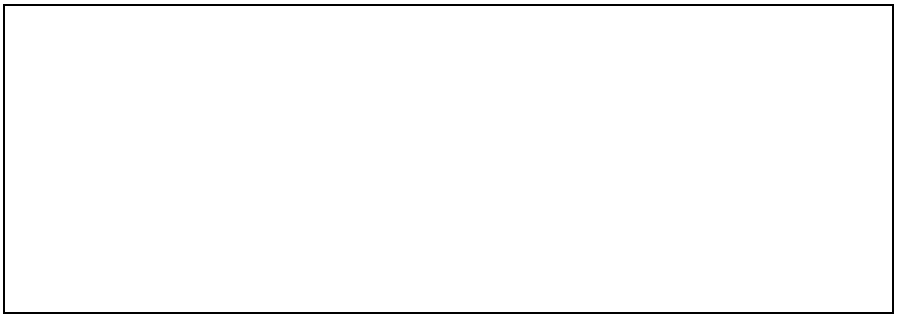


Figure 26. How to view the affect of Inconsistencies.

Figure 27 gives a complete decision tree for understanding how the three decisions affect the choose of a given piece of software. Clearly, the subjects get a first impression before actually gaining an understanding of the perceived or actual performance. At any point the opinion van be good or bad. If the path taken is “bad” then a subject could immediately discard the application and look for another method to solve the problem. IF the path is “good” then the user would continue to use the application, while building more opinions and potentially gathering actual performance. Of course, it would be useful to know the results from objective measures, but they are rarely available, and when they are they might not apply to the specific conditions.

The best result would be when all three paths taken were good, while the worst being all three paths taken were bar (if a subject would make it that far). The author feels that subjective opinions and perceived opinions would be more relevant to the end user and thus one could view the six possible outcomes from left to right as being in the order of usability. The numbers in the bottom right corner of each final outcome reflects this order. It is preceded by a “U” for User.

On the other hand, a manager might be more concerned with the bottom line and the actual performance measures so that it would take priority over the other four outcomes. Although even a manager would probably want the subjective opinions to carry more importance than the perceived objective importance since the manager would already know the actual performance data. This order is reflected in the number in the top left corner of each final outcome preceded by the letter “M” for management.

Others involved in the design, purchase, or use of software might take a different approach depending on the objectives of the application. Without consulting with potential end users one will always run the risk of making a poor selection. Understanding the users knowledge, skills, and abilities is essential to picking the appropriate application. If decisions are made without considering the actual users then information like actual objective performance from another

setting might not only not transfer in a similar manner but might cause additional complications like fatigue, low productivity, or even users quitting their jobs. It is sometimes difficult enough to initiate change in an environment without having complications from users acceptance on new systems. Working with users and testing out potential software to insure consistency and productivity will help all concerned in the long run.

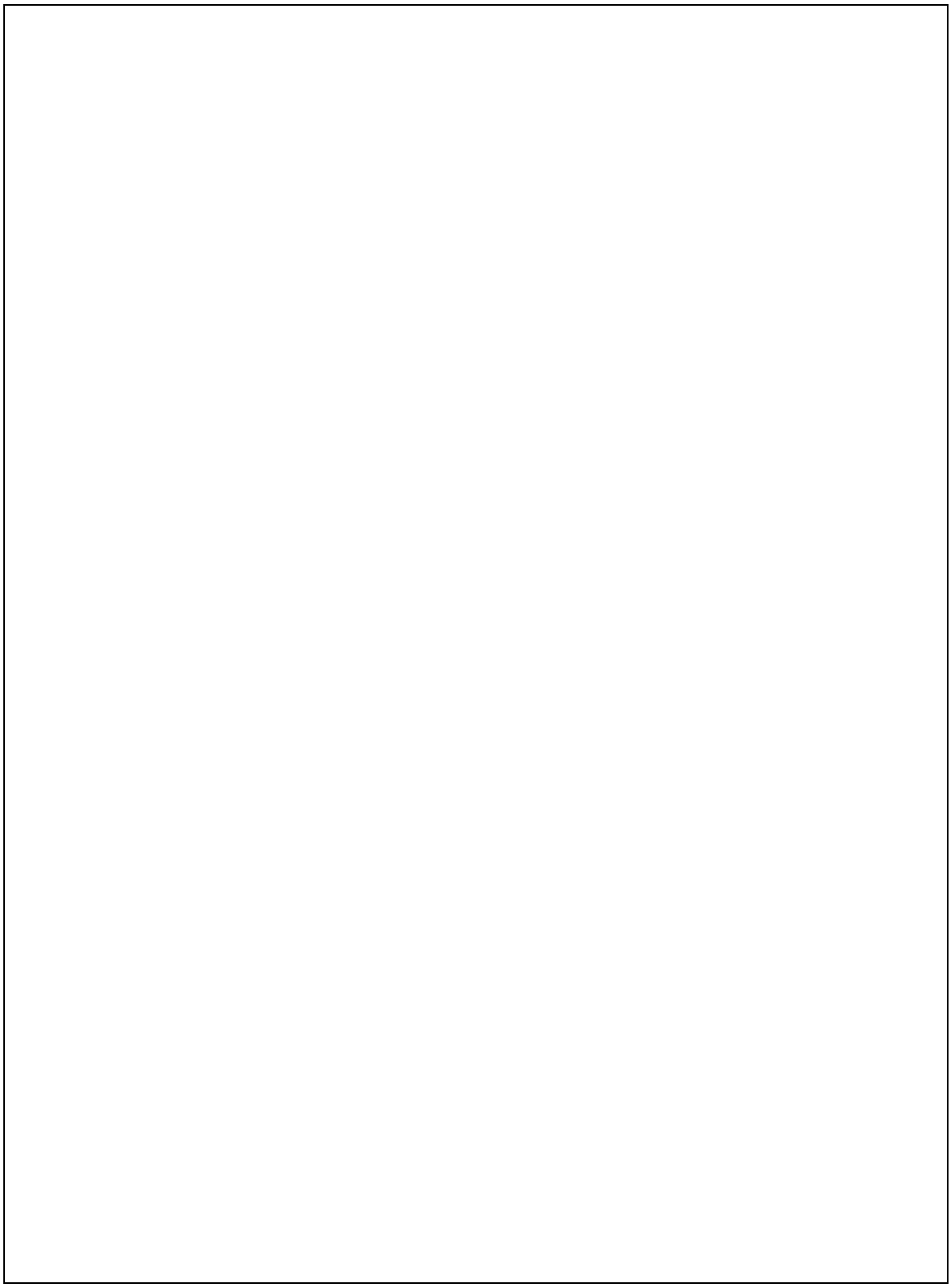


Figure 27. Flowchart for Software Evaluation.

CHAPTER 6
CONCLUSIONS

Conclusions from the analysis of the objective and subjective data were not consistent. If a true representation of a users ability to deal with inconsistent interfaces is needed then one needs to perform both subjective and objective measures. The data supported these conclusions;

- Subjective and Objective measures differed in their ability to discriminate between interfaces will varying degrees of inconsistencies.
- Subjects performance quickly recovered from using the inconsistent interface.
- Subjects were significantly more descriptive of inconsistencies than consistencies.
- Subjective measures reflected more emphasis on visual than low level procedural aspects of the interface.
- Subjective ratings support a strong overall preference for the consistent control version of the application over one inconsistent version even though the number of inconsistencies seeded in the inconsistent versions were almost identical.
- The type of inconsistencies impact objective and subjective measures differently.

- Subjects recognized more inconsistencies in a visually distinct version of the interface than one that was similar to the control version while showing significantly *better* objective performance.
- Subjects were better at keeping track of time when exposed to different versions of the same application.
- Subjects could discriminate inconsistencies that were vastly different than inconsistencies that were subtile.
- Subjects found many aspects of the interface to be related to the consistency of the interface design including that the application was; irritating, complicated, difficult to learn, frustrating, etc . . while not finding inconsistencies to relate to screen clutter, completeness, speed, or interpretability.
- Subjects found it easier to find physical layout, semantic, and virtual device inconsistencies. Few subjects noticed problems in the feedback and syntactic levels.
- Subjects were robust in finding and defining inconsistencies with no "formal" definition of consistency.

The observational data was useful in defining implication for the design of graphical user interfaces. The creation of potential guidelines and theories of human computer interaction was not the original intent of this research, but it provided the most robust results. Readers are referred to Section 5.4 for definitions of the terms used in the conclusions. Although most are specific to certain conditions, other are applied in general terms. Some can be used as confirmation of existing rules that have been

applied in other settings. Others are reiteration of existing guidelines but are given as further evidence of their appropriateness in UID

☐ If a visual frame is used in one section of a multi-screen application it should be consistent between screens, especially when the user can use it to position the cursor to activate buttons or enter fields.

☐ Dialog boxes which cause catastrophic errors should not contain the same placement and layout as other dialog boxes.

☐ All objects used on a screen display should either be hot, when appropriate, or appropriate feedback should be applied to distinguish cold title labels from hot multifunction fields and icons.

☐ Edit keys should be supported in all dialog boxes.

☐ Modal dialog boxes should be reserved for very important information or actions, and should not allow menu activation.

☐ Non modal (movable) dialog boxes should be provide when information on the screen will be obscured by the dialog box.

☐ A palette should be provided with menu bar equivalents. The option to reorient and move the bar should always be available.

☐ Iconic pop-up menus should follow the same rules as pop-up menus.

☐ When in programs than have multiple windows the tile command and window names should be provided in a menu.

 Some form of a clipboard message bar should be provided for users who frequently use the clipboard

 Certain types of prompt boxes should be centered in the work area. A consistent user defined approach should be used in all cases.

 The return icon should be kept visually separated from arrows that use a linear motion algorithm.

 Users should be prevented from changing information in infrequently editable fields without confirmation when the probability of them committing an error is high.

 Help should be provided in a consistent manner across applications. Context sensitive and command help should be available. Keyword searching should be provided.

 Novice user levels should be provided which will assist the user in performing operation. It should be automatically available, able to be turned off by users, and provide additional feedback without disrupting the current work flow.

 Interface objects that are unique to the application should be consistent with the users expectations within the application.

 Creating consistency with similar objects in other applications will improve users opinion of the interface and increase their acceptance.

 Consistency should not be applied to complete a overlying rule structure if it does not apply to the current function. That feature should not be available.

- [ ] Font status information should be provided in applications that have frequent style changes. It should be available on the users request.
- [ ] Screens that are very similar but can be misinterpreted easily by not noticing the specific changes should be made more distinct to retard users from making errors.
- [ ] Messages should be as specific as possible and should not interrupt the users work when they are not of paramount importance.
- [ ] The style of navigation should be matched to the appropriate navigation aid. Novice and expert users would both be accommodated (icon clicking and keyboard/arrow key navigation for lists).
- [ ] Keyboard presses should be properly stored with their modifier keys or they should not be stored at all.
- [ ] Palettes and corresponding menus and function keys should be arranged in a consistent manner. Navigation metaphors which rely on a left right orientation should be oriented in a similar fashion when first presented to the user.
- [ ] Feedback in a message bar should be provided in highly interactive systems. Dialog boxes should be reserved for very important information.
- [ ] Find searches should provide feedback to the specific actions taken when find strings were “not found”. The subject should be able to create a single list of all found text, with the ability to navigate to each occurrence or manipulate them all at once.
- [ ] Navigation by other than linear well understood movements in the computer world should be investigated before being applied. A suitable

metaphors should be provided to differentiate single and end motions from jumps of other than 1 “page” at a time.

Does consistency mean that developers and users will be restricted to having more productive software in the future? The answer is not simple, but seems to be no. Consistency allows users to move to new software with easy, without consistency users would be less inclined to more to new software, and would purchase less software, thus decreasing developers need for creating new software. One might reply that inconsistency has not kept IBM PC based developers from writing over 50, 000 applications. But one can only imagine how many more would have been written if subjects were more inclined to buy new software. From personal experience, the author spent a large amount of time making the decision to switch from Word to Word Perfect, but now that the author uses a Macintosh, it is not only simple to move from word processor to word processor, but to use both Nisus and Word 4.0, simultaneously. In addition, users of the Macintosh buy and use more software than the average IBM user.

By making the standards readily available and punishing developers who strayed from the spirt of the guidelines (by end users not purchasing the software) developers can spend more time creating innovative software that is not covered by the guidelines, thus extending the functionality and usability of software.

CHAPTER 7
FUTURE RESEARCH

The two most important issue that still remains unanswered; creating a rule base for assessing the impact of inconsistencies at different levels of an interface (i.e, Robert's levels, 1988), and the creation of a taxonomy to classify inconsistencies "consistently". This experiment explains when consistency should be applied at one level at the expense of another. It does not generalize those few observations to a global metric applicable across platforms and applications. A true taxonomy, than just a classification scheme, requires a rule base that can separate the issues presented to designers of graphical user interfaces. A command language (Moran, 1981) or user action notation (Hartson, 198? - no cite) requires a detailed understanding of the lowest level building blocks. This in itself might hamper its use by developers working under severe time restraints. Again the best solution, at this time, would be to create guidelines with instructions on their application that developers can apply without needing complicated flow diagrams and nebulous command notations. This might be fostered by not only increasing the number of existing guidelines (even though no one likes having even more guidelines to follow), but by giving designers the understanding of how to extend the influence of existing guidelines without having to do the research themselves. This means that companies or groups that propose standards have to be willing to create living documents than can be enhanced and extended quickly. Research into how to extend the quick keys would be a good effort. Some keys are already defined, while one could investigate creating a second tier of command sets that apply to specific type of applications (Word processing, graphics, database, etc…).

The long terms effects of inconsistencies were not explored in this research. If subjects were exposed to the inconsistent versions of the interface for an extended period then the effect of transfer might be more severe. Although the subjective opinions were pronounced, timed performance measures showed quick recovery from the presentation of the inconsistencies. It would be interesting to see how inconsistencies in virtual devices affect performance. Extensive literature is available on the use and functionality of real input devices like mice, trackball's, and touchscreen's, but the virtual devices like pop-up menus, scrolling lists and pull down menus need to evaluated to determine what types of information are better suited to a specific device. In general, the rules and understanding of when a pull-down menu is appropriate can be found in guidelines like Smith and Mosier (1984), but these do not address if other virtual devices would provide a better medium to present the information.

The topics discussed in Section 5.4 (Observational Data) provides a wide range of ideas based on observational data. One can take these issues and use them as a basis for creating experiments suited to analyzing these topics. The present researches shows coming extends to not being able to support individual inconsistencies as having effects on performance since there were usually dozens of variables that were being manipulated simultaneously. Issues concerning the users locus of control, understanding of current typestyles, creating hot objects, prompt box locations, adaptive interfaces, help, pop-up iconic menus and movement by multiples of more than one lend themselves to a controlled experimental setting.

As with any research more questions arose then were answered. Due to the global nature of this research, many low level questions were not addressed. Clearly there is still a need to answer those questions, and many new problems will begin to arise as the graphic user interface becomes a more common item in everyday society. Today even supermarkets have innovative touch screen, graphic interfaces to help customers locate items in the store. At the other end of the spectrum, SY-45, a document in development by the Human Interface group at McDonnell-Douglas, deals with the interface guidelines that all Space Station Freedom computer software and hardware must follow. The graphic interface used in the space station is a conglomeration of the current crop of GUI's. Many issues found in the supermarket setup and on the space station can be resolved

by understanding user expectations of consistency. Of course, the goals of the two user groups differ, as does the criticality of the information, but the low level research used to develop the supermarket system can be found to trickle up to the space station system and vise versa. The systems are more similar than dissimilar and by continuing to focus on the overall perspective and the low level topics a common middle ground can eventually be reached.

REFERENCES

Allwood, M.C. (1986). Novices on the computer: a review of the literature. *International Journal of Man-Machine Studies*, *25*, 633-658.

Apple Computer, Inc. (1987). *Human Interface Guidelines: The Apple Desktop Interface*. New York: Addison-Wesley.

Apple Computer, Inc. (1989). *HyperCard Stack Design Guidelines*. New York: Addison-Wesley.

Bachman, R. (1989). A methodology for comparing the software interfaces of competitive products. *Proceedings of the Human Factors Society 33rd Annual Meeting* (pp. 1214-1217). Santa Monica, CA: Human Factors Society.

Barnard, P.J., Hammond, N.V., Morton, J., Long, J.B., and Clark, I.A. (1981). Consistency and compatibility in human-computer dialogue. *International Journal of Man-Machine Studies*, *15*, 87-134.

Berry, R.E. (1988). Common user access- A consistent and usable human-computer interface for the SAA environments. *IBM Systems Journal*, *27*(3), 281-300. Armonk, NY: IBM.

Blake, T. (1986). *Introduction to the Art and Science of User Interface Design*. Intuitive Software and Interactive Systems, CA.

Bouchard, J.C., and Robert, J.M. (1989, September). *Towards a consistency manager for user interfaces: Requirements and constraints*. Poster session presented at HCI International '89, Montreal, Quebec.

Bowers, V.A. (1990). *Concurrent versus retrospective verbal protocol for comparing window usability*. Unpublished doctoral dissertation, Virginia Polytechnic Institute & State University, Blacksburg, Va.

Briggs, P. (1989). Consistent benefits for the system designer and the end-user. *Applied Ergonomics*, *20*(3), 160-167.

Brown, C. M. (1988). *Human-Computer Interface Design Guidelines*. Norwood, New Jersey: Ablex.

Card, S.K., Moran, T.P., and Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Carroll, J.M., Mack, R.L., and Kellogg, W.A. (1988). Interface metaphors and user interface design. In M. Helander (Ed.), *Handbook of Human-Computer Interaction*. Amsterdam: Elsevier Science Publishers, B.V.

Chin, J.P., Diehl, V.A., and Norman, K.L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *CHI '88 Conference Proceedings, Human Factors in Computing Systems* (pp. 213-218). NY, NY: ACM.

Cohen, M., and Schirmer, E. (1989, September). *Human factors in the success of a commercial software package: Lotus 1-2-3*. Paper presented at HCI International, Boston, Ma.

Coleman, W.D., Williges, R.C., and Wixon, D.R. (1985). Collecting detailed user evaluations of software interfaces. In *Proceedings of the Human Factors Society 29th Annual Meeting* (pp. 240-244). Santa Monica, CA: Human Factors Society.

Digital Equipment Corporation. (1988, December). *XUI Style Guide*. [Computer manual]. Maynard, Mass: Author.

Gomoll, K, and & Nicol, A. (1990, January). Note #2 Design Principles for On-Line Help Systems. *Human Interface Notes*, Apple Computer Inc.

Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, *32* (10), 1164-1173.

Grudin, J., Ehrlich, S.F., and Shriner, R. (1987). *Positioning human factors in the user interface development chain*. Manuscript submitted for publication.

Ham, M. (1987). Software design rules. *Dr. Dobb's Journal*, (7), 112-116.

Hammer, M., Kunin, J.S., and Schoichet, S. (1983). What makes a good user interface? *Office Automation Conference* (pp. 121-130). Washington: AFIPS Press.

IBM (1987, First Edition). *Common User Access Panel Design and User Interaction*. [Computer Manual]. Boca Raton, Florida: Author.

Jensen, S. (1990, January). Note # 4 Movable Modal Dialog boxes. *Human Interface Notes,* Apple Computer Inc.

Kellogg, W.A. (1987). Conceptual consistency in the user interface: Effects on user performance. In H.J. Bullinger and B. Shackel (Eds.), *Human Computer Interaction - Interact '87, Proceedings of the IFIP Conference* (pp. 389-394). NY, NY : Elsevier Science Publishers B.V.

Kieras, D.E., and Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Language*, *25*, 507-524.

Kieras, D.E., and Polson, P.G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, *22*, 365-394.

Koritzinsky, I.H. (1989). New ways to consistent interfaces. In J. Nielsen (Ed.), *Coordinating User Interfaces for Consistency* (pp. 93-106). New York: Academic Press, Inc.

Mayer, Richard E. (1981). The psychology of how novices learn computer programming. In B. Curtis (Ed.) *Tutorial: Human factors in software development*. (Reprinted from *Computer Surveys* , 1981, *13* (1), 121-141.

Mish, F.C. (1983). *Webster's ninth new collegiate dictionary*. Springfield, Massachusetts: Merriam-Webster, Inc.

Moran, T.P. (1981). The command line grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, *15*, 3-50.

Paap, K.P., and Hofstand, R.J.R. (1986). The optimal number of menu options per panel. *Human Factors*, *28*(4), 377-385.

Payne, S.J., and Green T.R.G. (1986). Task-action grammars: A model of the mental representation of task languages. *Human Computer Interaction*, *2*, 93-133.

Polson, P.G., Muncher, E., and Engelbeck, G. (1986). A test of common elements theory of transfer. In *CHI '86 Conference Proceedings, Human Factors in Computing Systems* (pp. 78-83). NY, NY: ACM.

Polson, P.G. (1989, December 11). Personal communication.

Polson, P. (1988). The consequences of consistent and inconsistent interfaces. In R. Guindon (Ed.), *Cognitive science and its applications for human-computer interaction* (pp. 59-108). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Reisner, P. (1981). Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions in Software Engineering*, *SE-7*, March, pp. 229-240.

Robert, J.M., and Wang, W. (1989, September). *A statistical method to evaluate the relative importance of inconsistencies in user interfaces and measure the benefits of solutions*. Paper presented at the 2nd International Scientific Conference on "Work with Display Units", Montreal, Quebec.

Robert, J.M. (1988). Detecting inconsistencies in user interfaces. Analysis, design, and evaluation of man-machine systems. *Selected Papers From the Third IFAC/IFIP/IEA/IFORS Conference* (pp. 357-363). Oslo, Finland: Man-Machine Systems.

Roberts, T.L., and Moran, T.P. (1982). Evaluation of text editors. In *Proceedings Human Factors in Computer Systems Conference*, pp. 136-141. NY, NY: ACM.

Shneiderman, B. (1987). *Designing the user interface*. Reading, Massachusetts: Addison-Wesley.

Siegel, S. (1956). *Nonparametric statistics for the behavioral sciences*. NY, NY: McGraw Hill.

Smith, S.L., and Mosier, J.N. (1984). *Design guidelines for the user interface for computer-based information systems*. Bedford, MA: MITRE Corporation, Electronics Systems Division. (Available from NTIS).

Tullis, T.S. (1983). The formatting of alphanumeric displays: A review and analysis. *Human Factors*, *25*(6), 657-682.

Virzi, R.A. (1990). Streamlining the design process: Running fewer subjects. In *Proceedings of the Human Factors Society 34th Annual Meeting* (pp. ???-???). Santa Monica, CA: Human Factors Society.

Voelcker, J., Wallich, P., and Zorpette, G. (1986). Personal computers: lessons learned. *IEEE Spectrum*, *23* (5), 44-75.

Whiteside, J., Jones, S., Levy, P.S., Wixon, D. (1985). User performance with command, menu, and iconic interfaces. In *CHI '85 Conference Proceedings, Human Factors in Computing Systems* (pp. 185-191). NY, NY: ACM.

Wolf, R. (1989). Consistency as process. In J. Nielsen (Ed.), *Coordinating User Interfaces for Consistency* (pp. 89-92). NY, NY: Academic Press, Inc.

APPENDIX A: Informed Consent Form

You have been solicited as a research participant for our evaluation involving the use of a new software application for the Macintosh computer. Prior to the actual session a vision test and Macintosh interface analysis needs to be performed. Because you have certain rights, this informed consent has been drawn-up to delineate those rights and responsibilities of both the research participants and the experimenters. Please read through this document before you decide whether or not to participant. The evaluation is being run by; **Dr. Robert Beaton**, Faculty Advisor (703-231-5936) and **Richard Miller**, Graduate Student (703-951-3496).

First you will be given a vision test. The purpose of the vision test is to determine whether your vision meets the criteria for participating in the evaluation. The vision test given to you is not a professional eye test; therefore, the results should not be considered an accurate description of your vision. A professional eye doctor should be consulted for an accurate examination of your vision.

The eye test consists of two parts. If you pass both tests you will be asked to work with a Macintosh application. Even if you are knowledgeable with the Macintosh you might have some difficulty working with the program. This is normal, as the program is not perfect. If you can finish eight out of the ten tasks you will be asked to participate in this evaluation. The application is called the "Personal Organizer" and includes such items as a datebook, telephone directory, and other office items.

These tests will take a total of fifteen minutes. You will not be paid for the vision and Macintosh test. However, if you do participate in the evaluation, you will be compensated for the time you spend in the evaluation.

There are no known risks associated with this evaluation. The only known discomfort would possibly be fatigue resulting from the length of the evaluation. Rest-breaks occur during the ninety minute evaluation. During the evaluation period you will be asked to use the application and rate the program using some short questionnaires.

You may choose monetary compensation or a set of shareware and freeware programs for you time and effort. Your rights as a participant are as follows:

1. You have the right to withdraw from the procedure at any time for any reason.

2. At the conclusion of your participation, you have the right to see your data and withdraw them if you so desire. If you decide to withdraw your data, please inform the administrator immediately. Otherwise, identification of your data will not be possible because they will have been collapsed with other data in order to ensure anonymity.

3. You are requested to refrain from discussing the evaluation with other people (especially people in your area, from where other evaluators might me drawn). We expect all data to be collected by the end of July, 1990. Following that date feel free to discuss the evaluation with anyone you wish.

4. If you wish to receive a summary of the results of this evaluation, please include you address (where you will be six months from now) with your signature below. If you should desire more detailed information about the evaluation, you may contact the above mentioned faculty or graduate student who will provide you with a full

report. They will also be able to answer any question you may have regarding your participation.

5. If you do not wish to contact the researchers, you may contact **Dr. E. R. Stout** who is Chairman of the Institutional Review Board. His telephone number is (703) 231-5281. Dr. Stout will be able to handle any complaints or concerns you have regarding the evaluation or your participation.

Finally, we greatly appreciate your invaluable time and effort for participating in this evaluation. Remember, you cannot fail any part of this session, there are no right or wrong answers. The session is to view problems with the software not with you. Your signature below indicates that you have read this in its entirety, and that you consent to participate in this evaluation. Are there any questions?

_____ _____

Name Address

_____

Signature

APPENDIX B: Macintosh Screening Test for Users

(1) Have you used ResEdit?

(1) Have you used StuffIt?

(2/2) Have you used HyperCard? - Have you written scripts in HyperCard?

(2/2) What menu choices are usually found in the EDIT menu? What are their quick keys?

(2) Where is HELP usually found in a program and how is it accessed?

(2) How do you delete a file that the Finder reports as being locked?

(2) Have you used a color Macintosh?

(2) What does the command “Find” usually do in an application? and how is it activated?

(2) How can you scroll through the contents of a window?

(3) Name three styles for fonts found in a typical word processor and their quick keys?

(2) How do you save changes made in HyperCard?

(2) Name two buttons found in an open dialog box?

(3) What is a floating palette? How do you show and hide palettes?

(2) What % transfer would you say that have for going from one program to another?

In general, what would you guess to be the quick keys for these commands

(4) Undo Cut Copy Paste Goto Help Find Close the window Cancel

(1-1) If you have a database program or any program that contains multiple fields what would pressing the TAB key do? what about shift-TAB?

(2-1) If you are in a modal Dialog box how do you copy text typed in a field in the box to the clipboard? What happens if you try to click outside of the box area?

(2) Assume you are given a program that simulates the reading of a large book with many chapters. Name some methods for navigating like using a menu choice to jump to the beginning of the last chapter.

(2) If I gave you a piece of software and did not tell you what it did, could you install it on the Macintosh and figure out how to start to use it?

(2) If I give you a new word processing program, how long would it take for you to figure out the program enough to be qualified to type out a simple resume? (not including the time to typing)

(1) On a scale of 1 to 10, 1 being a complete novice and 10 being an expert in using Macintosh programs, what would you give yourself as a rating?

(2) What does a pop-up menu do, and how can you recognize one?

Total Score is out of a possible 50 points. Questions were asked verbally for all subjects, to allow for scoring to be flexible.

APPENDIX C: Post-test Questionnaire

Please rate the interface you just used on these scales. Circle a number to represent your impressions of what you feel is appropriate. Note: The scales do not always go from “Good” to “Bad” (left to right), so please read each pair carefully!

The interface refers to the objects, actions, and methods used to work with the application. The interface includes every action you perform and input you make while using the application.

The interface was

Pleasing						Irritating	
1	2	3	4	5	6	7	
Unfriendly							Friendly
1	2	3	4	5	6	7	
Complete							Incomplete
1	2	3	4	5	6	7	
Cooperative							Uncooperative
1	2	3	4	5	6	7	
Undependable							Dependable
1	2	3	4	5	6	7	

Complicated Simple

1 2 3 4 5 6 7

Inconsistent Consistent

1 2 3 4 5 6 7

Natural Unnatural

1 2 3 4 5 6 7

Intelligent Unintelligent

1 2 3 4 5 6 7

Uninterpretable Interpretable

1 2 3 4 5 6 7

Fast Slow

1 2 3 4 5 6 7

Adaptive Unadaptive

1 2 3 4 5 6 7

Useful (functions and commands) Useless

1 2 3 4 5 6 7

Redundant (functions)							Concise
1	2	3	4	5	6	7	
Cluttered (Screen layout)							Uncluttered
1	2	3	4	5	6	7	
Unsafe (Data security)							Safe
1	2	3	4	5	6	7	
Maintainable							Unmaintainable
1	2	3	4	5	6	7	
Easy to learn							Difficult to learn
1	2	3	4	5	6	7	

The application consists of the interface and the underlying functionality and processing that creates the information available to you.

The application made me feel

Not frustrated							Frustrated
1	2	3	4	5	6	7	
Unproductive							Productive
1	2	3	4	5	6	7	

that it is not worth buying | that it is worth buying

1 2 3 4 5 6 7

Relaxed | Tense

1 2 3 4 5 6 7

Optimistic | Pessimistic

1 2 3 4 5 6 7

Competent | Incompetent

1 2 3 4 5 6 7

Pleased | Disgusted

1 2 3 4 5 6 7

Bored | Engaged

1 2 3 4 5 6 7

Unhappy | Happy

1 2 3 4 5 6 7

Intelligent | Stupid

1 2 3 4 5 6 7

Pleasant | Annoyed

1 2 3 4 5 6 7

Note: Actual Questionnaire had more space for answers

What about the interface did you like?

_____

What do you think we should change in the interface in future revisions? (What could be changed in this program that would make it more “Mac-like”)

_____

Can you think of any addition features or functionality that should be added to the application or the interface, if so what?

_____

APPENDIX D: Post-Experiment Questionnaire

Note: Actual Questionnaire had more room for free response answers.

Please ask the investigator to open the second version of the application.

Did you find the second program easy to learn?

Yes, very much — No, not at all

1 2 3 4 5 6 7

When you worked with the program was it consistent with how you expected it to act?

Yes, very much — No, not at all

1 2 3 4 5 6 7

Do you think if the interface in the second program was more consistent it would have been easier to use?

Yes, very much — No, not at all

1 2 3 4 5 6 7

Did the inconsistencies in the second version effect the time it took to complete the tasks?

Yes, very much — No, not at all

1 2 3 4 5 6 7

How many minutes do you think it took to complete each of the six sessions?

1 _____ 2 _____ 3 _____ 4 _____ 5 _____ 6 _____

How would you compare the third version of the program to the second?

Better Second Version Worse

1 2 3 4 5 6 7

Confusing Second Version Understandable

1 2 3 4 5 6 7

Easier Second Version Harder

1 2 3 4 5 6 7

More Consistent Second Version Less Consistent

1 2 3 4 5 6 7

Simpler Second Version Complicated

1 2 3 4 5 6 7

What, in general, did you like about the second program you used?

_____

What, in general, did you **not** like about second program you used?

_____

Overall, how would your compare the three versions of the program?

First & Second:	Equal						Different
	1	2	3	4	5	6	7
Second & Third:	Equal						Different
	1	2	3	4	5	6	7
First & Third:	Equal						Different
	1	2	3	4	5	6	7

Try to compare the Consistency or Inconsistency (circle one) of the second versions features to the other two versions as well as other Macintosh and non-Macintosh programs that you have used.
Example:

The Undo Command (Con - Incon) Why? <u>It would not let me undo my most recent command and it used a different function key than the standard Apple programs</u>

1. Menu Structure (Con - Incon) Why? _____

2. Icon Placement (Con - Incon) Why? _____

3. Edit Keys (Con - Incon) Why? _____

4. Menu items order (Con - Incon) Why? _____

5. Help system (Con - Incon) Why? _____

6. Navigation arrows (Con - Incon) Why? _____

7. Icons for the Functions (Con - Incon) Why? _____

8. 'Find' within a function (Con - Incon) Why? _____

9. 'Find' between functions (Con - Incon) Why? _____

10. Function Key layout (Con - Incon) Why? _____

11. Phone Setup input (Con - Incon) Why? _____

12. Dialing a number (Con - Incon) Why? _____

13. Error and other Messages (Con - Incon) Why? _____

14. Auditory feedback (Con - Incon) Why? _____

15. Visual feedback (Con - Incon) Why? _____

16. Clock function (Con - Incon) Why? _____

17. Wording (uppercase/lowercase etc…) (Con - Incon) Why? _____

18. Using Tab keys to navigate (Con - Incon) Why? _____

19. Color (Con - Incon) Why? _____

20. Floating Palettes (Con - Incon) Why? _____

21. Function specific actions (Sorting, Find, etc…) (Con - Incon) Why? _____

22. Pop-up Menus/ Icons (Con - Incon) Why? _____

APPENDIX E: General Questionnaire

If you need additional space please ask the investigator for more paper. The more you say the better the results of this study.

What is your Age? _____ Male or Female? _____ Department ? _____

Faculty - Staff - Graduate Student - Undergraduate (circle one). If a student, what year?__

How many years of experience with all computers have you had? _____

How many years of experience with Macintosh computers have you had? _____

What kind of computers do you use? _____

What kind of computers do you own? _____

About how many hours per week do you use a Macintosh computer? (circle one)

0-4 hrs/wk 4-8 hrs/wk 8-12 hrs/wk 12-16 hrs/wk
16-20 hrs/wk 20-24 hrs/wk 24-28 hrs/wk more than 28 hrs/wk

In general, do you like working with computers?

Yes, very much No, not at all
1 2 3 4 5 6 7

Do you find most programs on the Macintosh easy to learn?

Yes, very much No, not at all
1 2 3 4 5 6 7

Why or why not?

APPENDIX F: Instructions for Users

Thank-you for participating in this study of computer usage. This research is done, in an effort to better understand the human-computer interface. You are being asked to be part of the development process of a computer based application. Your thoughts, comments, and actions will help shape the development of the system.

There will be two sections to this experiment, and you will take approximately two hours to complete the tasks. The first section is a screening and you must pass this screening in order to participate in this experiment.

[Administer Screening before continuing with instructions / Set Keyboard Speed]

To Live in a World of Love is to Live in a World of Peace

What you do today might have been done better yesterday

This is a test, as life is a test

I am the color ??
colors I am the color ??
I am the color ??
I am the color ??
I am the color ??

Note: These text strips were presented in the Red, Green, Blue, Yellow, and Black, respectively.

Please Type Here..

[If they do not pass the tests, thank the participant and excuse them from the evaluation.]

Remember all data is kept strictly confidential, and there is no way for the experimenter to separate your data from previously taken data after the completion of your session. The VCR camera behind you shoulder is set to record your use of the computer and to help record you comments made during an interview at the end of the session.

Please look at the program presented on the computer, it is called The Personal Organizer. The computer will prompt you with a single instruction in a separate window. Please follow the instructions and complete the task as outlined. If you cannot figure a way to complete a task inform the instructor. Try your best to complete each and every task in a timely manner. Please verify that you have completed everything that the instructions ask for before continuing to the next instruction. It is more important that you complete each and every instruction that to go fast. Use the HELP system if you are having problems. In many cases the HELP system will be faster that randomly experimenting with the interface. Once you start a new task you will not be allowed to go back to the previous task. Remember, you are assisting in the program development process. Once you have completed the task you will need to return to the main screen and click the “DONE!” button. You return to the main screen by clicking the Main Screen button in the application. Clicking “Done!” will end the trial and start the next trial. After 8 trials you may take a break. There will be six sets of trials and you may take a break after each set of trials.

Here are some rules to follow during the experiment:

- Do not try to quit the program or switch out of the application using Multifinder,
- Try to use the Tab keys and Quick keys to speed your performance,

• Do not use the message bar, you should be able to perform all tasks without typing any commands. Use the menus, icons, quick keys, function keys and arrow keys to perform actions,
• Unless the instructions specially request a certain font or style, do not waste time making text match other text styles when editing,
• Searching and Navigation is very important to completing the instructions. If you find yourself clicking arrow keys over and over again to move around, try using the “Find” function or another use a different method to move around the quickly. Look for pop-up menus.
• Please read over the list of quick keys that is provided, this will give you an idea of some of the most common function and actions,
Are there any questions? Are you ready to begin?

[Administer General Questionnaire]
[Start program]
[Once the Tasks are completed]
[Rest Break ? Repeat for another set of instructions]

Please answer the questions on the questionnaire provided. This will take a few minutes. Please try to consider each question carefully, your expert opinions are very valuable.
[Administer the Questionnaire]

[Repeat next two sections for the last two sets of tasks]

Please continue to use the interface by following the instructions in the window. The sets of tasks used in this interface are similar to those provided in the first test. Please try to complete each and every task in a timely manner. Note the new list of quick keys, and the new keyboard template.
[Once the Tasks are completed] [Rest Break]

Please answer the questions on the questionnaire provided. This will take a few minutes. Please try to consider each question carefully, as they are not the

same as found in the first questionnaire. Please base you decisions on the last two sets of instructions.
[Administer the Questionnaire]

Thank-you for participation in this study. You have completed using all three interfaces. Please take some time to complete this final questionnaire.
[Administer Final Questionnaire]

APPENDIX G: Seeded Consistencies and Inconsistencies

Each version of the application had a set of characteristics that were defined by the experimenter as inconsistencies. Most of the inconsistencies defined in the two inconsistent versions were “inconsistent” with the control version of the interface. Each area that was known by the experimenter as to be possibly defined by subjects as an inconsistency was placed into one of twenty-two topics. These topics are identical to the twenty-two topics subjects used during the evaluation of the treatment version of the interface. As discussed in the Discussion section, some of the inconsistencies do not fit “exactly” into the topic. Inconsistencies that were not good fits were placed in the topic that was related.

This section in conjunction with the screen dumps in Appendix J can be used to gain an understanding of how the application worked and looked. It is difficult to explain many of the inconsistencies without looking at the menus, icons, layouts, lexicon, and devices used in the three interfaces.

Every inconsistency in this section was categorized into one of the eight levels of the classification scheme. The letter after the explanation of the inconsistency explains where the item was classified. Items were only classified into one category even if the item could apply to numerous categories. One should review all items before determining that an item might be classified in the “wrong” category. Some items might appear in slightly different forms in different topics and with different categories. Small changes in how the item is discussed can play an important role in its categorization. Please see the example of the Sort button, in Table 35, for clarification on how syntax can impact the classification scheme.

APPENDIX H: Tasks Performed By Subjects

The text for each of the 48 tasks used in this experiment are presented here. The Block number and task type referenced at the end of each task can be compared to the general tasks explained in Table 4. (Block #-Task Type)

1-A. See if you are available for dinner on August 21, 1990 in the Weekly Planner. If you are not available reschedule what is there for the same time in the previous week. Then put down “Dinner with the Stevens family“ for that day. •

1-B. Find Alan Chao’s telephone number and put his name and lab number with “Call about weekly parts supply” in the Weekly Planner on September 7 and 14, 1990. Do not go to the Address Roledex to find this information! •

1-C. Go to the Phone Setup function and add “724” to the list of prefixes, then change the Long Distance code to “28” (Please keep a space between the prefixes). •

1-D. Delete the last two pages of the To Do function (even if they are empty). Put “Birthday Party List” in 14 pt italic type at the top of the visible page on the right. Put “Jamie, Karen, Steve, and Rob” under it in 14 pt. plain type. Insert this information before anything on the current page. •

1-E. Go to “Kentucky” (area code 606) in the Area Code function. Place “Baseline, Sweet Briar, Waterpoint” in the list of locations, in their correct alphabetical order. •

1-F. Delete (not Erase) the cards in the Address Roledex for Ira Ungermann and Steve Jones, and Royce Walthrop. •

1-G. Find the meeting with Jake Burns in the Weekly Planner. Verify that the name of his company is correctly identified as compared to what is in the Address Roledex. Correct the Planner entry (not the Address Roledex), if necessary. •

1-H. Get Help for the “Sort” icon in the Address Roledex function. You will verify that you are getting Help for the correct function by reading the information out loud. Do not actually perform a “Sort”. •

2-I. Add “Meeting with Boss at noon” in bold text for October 5th and 19th, 1990 in the Weekly Planner. •

2-J. Find the area code for Carmel, California in the Area Code function. Put it in front of both telephone numbers on the two address cards in the Address Roledex that show Carmel, California for an address. •

2-K. Go to the last card in the Address Roledex and sort the cards by last name. •

2-L. Go to pages 5 and 6 of the To Do function and add two more pages. Put “Phase 2 work group” in 14 pt underline type at the top of the visible page on the right. Put “Scully, Jobs, Bush, Miller, Swede, Willis, Jones” under it in 14 pt. plain type. •

2-M. Add a card to the Address Roledex with “George Bush White House Washington D.C. 20345”. The first telephone field should have “345-0567”, with the ’Location’ button set to “OFFICE“. •

2-N. Dial Major Greer’s lab number with the speaker setting from the Weekly Planner page with his name on it. Do Not go to the Address Roledex. •

2-O. Find the meeting in Montreal with Robert Stevens in your Weekly Planner. Make sure his Montreal phone number has the correct area code, as

compared to what is found in the Area Code function. Correct the Planner entry, if necessary. •

2-P. Get Help for the 'Toll Call' option from within the Phone Setup function. You will verify that you are getting Help for the correct function by reading the information outloud. Do not actually change the 'Toll Call' option. •

3-A. See if you are available for lunch on September 13, 1990 in the Weekly Planner. If you are not available reschedule what is there for the same time in the following week. Then put down "Lunch with Apple Computer" for that day. •

3-B. Find Betty Carr's telephone number and put her name and office phone number with "Phase-2 Overview Meetings" in the Weekly Planner on October 3 and 10, 1990. Do not go to the Address Roledex to find this information! •

3-C. Go to the Phone Setup function and add "725" to the list of prefixes, then change the Outside Line to "9" (Please keep a space between the prefixes). •

3-D. Delete the last two pages of the To Do function (even if they are empty). Put "Mac User Group" in 14 pt bold type on the visible page on the right. Put "Jay, Sarah, Barbara, Irwin" under it in 14 pt. plain type. Insert this information before anything on the current page. •

3-E. Go to "Georgia" (area code 404) in the Area Code function. Place "Cobb, East Point, and Vinnings" in the location list, in their correct alphabetical order. •

3-F. Delete (not Erase) the cards in the Address Roledex for Jerry Garcia and Roy Swain, and Karl Daly. •

3-G. Find the meeting with Andy Cohen in the Weekly Planner. Verify that the name of his company is correctly identified as compared to what is in the Address Roledex. Correct the Planner entry (not the Address Roledex), if necessary. •

3-H. Get Help for the “Phone Setup” icon in the Weekly Planner function. You will verify that you are getting Help for the correct function by reading the information outloud. Do not actually use the “Find” operation. •

4-I. Add “Ski with club if there’s snow” in italic text for January 6th and 20th, 1991 in the Weekly Planner. •

4-J. Find the area code for Mobile, Alabama in the Area Code function. Put it in front of both telephone numbers of the two Address Roledex cards that show Mobile, Alabama for an address. •

4-K. Go to the last card in the Address Roledex and sort the cards by last name. •

4-L. Go to pages 5 and 6 of the To Do function and add two more pages. Put “Groceries for Party” in 18 pt bold type at the top of the left visible page. Put “Grapes, Milk, Cheese, Chips, Coke, Beer, and Eggs” under it in 18 pt. plain type. •

4-M. Add a card to the Address Roledex with “Kay Whitmore Eastman Kodak Company Rochester, NY 14653”. The first telephone field should have “724-5150”, with the ’Location’ button set to“OFFICE”. •

4-N. Dial Dorris Schwartz’s Office number with the modem (tone) setting from the Shopping List Page of the To Do list’s. Do Not go to the Address Roledex. •

4-O. Find the meeting and phone number for Sierra Capitol in your Weekly Planner. Make sure the phone number has the correct area code, as compared to what is found in the Area Code function. Correct the Weekly Planner entry, if necessary. •

4-P. Get Help for the “Dial Number” icon in the Area Code function. You will verify that you are getting Help for the correct function by reading the information that is presented outloud. Do not actually bring up the “Dial” window. •

5-A. See if you are available for dinner on August 28, 1990 in the Weekly Planner. If you are not available reschedule what is there for the same time in the following week. Then put down “Dinner with the In-laws” for that day. •

5-B. Find Ted Holt’s telephone number and put his name and lab number with “Schedule group meeting” in the Weekly Planner on September 3 and 10, 1990. Do not go to the Address Roledex to find this information! •

5-C. Go to the Phone Setup function and add “726” to the list of prefixes, then change the International code to “011”. (Please keep a space between the prefixes). •

5-D. Delete the last two pages of the To Do function (even if they are empty). Put “Basketball Team” in 14 pt italic type on the visible page on the left. Put “Me, John, Susan, Gloria, Jack” under it in 14 pt. plain type. Insert this information before anything on the current page. •

5-E. Go to “Florida” (area code 813) in the Area Code function. Place “Bradington, Coconut Grove, Willmount” in the location list, in their correct alphabetical order. •

5-F. Delete (not Erase) the cards in the Address Roledex for Dean Diffie and Beverley Richie, and Glenn Kelley. •

5-G. Find the meeting with Steve Arronan in the Weekly Planner. Verify that the name of his company is correctly identified as compared to what is in the Address Roledex. Correct the Planner entry (not the Address Roledex), if necessary. •

5-H. Get Help for the “Extend Planner” icon in the Weekly Planner function. You will verify that you are getting Help for the correct function by reading the information outloud. Do not actually “Extend” the planner. •

6-I. Add “Meeting with Copier team” in underline text for January 4th and 18th, 1991 in the Weekly Planner. •

6-J. Find the area code for Winchester, Virginia in the Area Code function. Put it in front of the both telephone numbers on the two Address Roledex cards that show Winchester, Virginia for an address. •

6-K. Go to the last card in the Address Roledex and sort the cards by last name. •

6-L. Go to pages 5 and 6 of the To Do function and add two more pages. Put “Movies to Rent” in 18 pt bold type at the top of the left visible page. Put “Die Hard, Mask, Batman, Star Wars, Animal House” under it in 18 pt. plain type. •

6-M. Add a card to the Address Roledex with “Jim Wilstern Eastman Kodak Company Rochester, NY 14653”. The first telephone field should have “726-9302”, with the ‘Location’ button set to “OFFICE”. •

6-N. Dial Dorris Schwartz’s Office number with the speaker setting from the Weekly Planner page with her name on it.Do Not go to the Address Roledex. •

6-O. Find the meeting with Bobby Heisman in your Weekly Planner. Make sure the phone number has the correct area code, as compared to what is found in the Area Code function. Correct the Weekly Planner entry, if necessary. •

6-P. Get Help for the ‘Extend Calendar’ icon in the Yearly (Six-Month) Calendar. You will verify that you are getting Help for the correct function by reading the information that is presented outloud. Do not actually “Extend” the calendar. •

APPENDIX I: Identification of Inconsistencies

TABLE I-1. Inconsistencies Mentioned by Subjects in Group 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	D					D		T				S		F	F					T		D P
2				F			LL			D		D	P					D			P	
3	F		T		SY						S		L	F								
4								D								T			T			D
5		P		TY			T						P									
6	D				F			T T								T						
7	T							M	M					F								
8																						
9			Y				L	F	F T	D			Y			T						
10		S			Y			T				Y										

1 1							L					T								F

Note: Group 1 received the control “consistent” version in Set 2.

Key of Inconsistencies Found:

Physical Location - P	Memory - M	Virtual Device - D	Semantic - S
Unavailable Task - T	Syntactic - Y	System Status (Feedback) - F	Lexical - L

TABLE I-2. Inconsistencies Mentioned by Subjects in Group 2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1					D		L	M	M										T	L		
2	P			T	Y			MT		P				F		Y				P		
3	T																		T			
4		P		P	Y	P		M	M	P	P	P		F			L		T			F
5	L	DP		P	SY			MT	M	P	Y	T	F	F	D		LL	D	F	LS	M	F
6	PP			LPS	S			M	M	PPT	D	TM	P				L	D	T	SY	T	F
7	PP	P	P			D		M		P	D				F	P			T	YD	M	F
8	Y		D												F	P			T			Y
9				P	D		L				D											P
10	PPL							T		P	DD						L					F
11	Y	P		P	P	P	P				D			F	F		LL		T	D		Y

Note: Group 2 received the treatment “inconsistent” version A in Set 2.

Key of Inconsistencies Found:

Physical Location - P Memory - M Virtual Device - D Semantic - S

Unavailable Task - T Syntactic - Y System Status (Feedback) - F Lexical - L

TABLE I-3. Inconsistencies Mentioned by Subjects in Group 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	DF	P						M			D	Y			F	Y		Y	S	Y	S	D
2	P	S			D		S												ST	Y	D	P
3				P	Y	P		T		P	P		L	F			L		T	D	D	
4	LD	P	P	PP	L		L	MT	M		D	T			S	T	L				T	YL
5		P		D		P		M	M				D				L		T	Y	DL	
6	Y	P		Y	P		P	M				D	P			P	S	D	T	D		P
7	S					P	LP													Y		F
8	D	P	S	Y	T	TP		T	T	TP	LP	T		F	L	T	LP	D	T	T	TT	
9	D	P	P	P		P			M		P	TS						Y	S		P	F
10	P	P		TP	D			T		P							L	Y		P		
11	P	PP	T	YYP	Y	P		M	M		D	SD	P	F		P		DY		Y	D	

Note: Group 3 received the treatment “inconsistent” version B in Set 2.

Key of Inconsistencies Found:

Physical Location - P Memory - M Virtual Device - D Semantic - S

Unavailable Task - T Syntactic - Y System Status (Feedback) - F Lexical - L

TABLE I-4. Number of Inconsistencies by Type and Question for Each Version.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
P/1		1											2								1	1
P/2	7	4	1	5	1	2	1			7	1	1	1			2				1		1
P/3	3	9	2	6	1	6	2			3	3		2			2	1			1	1	2
M/1								1	1													
M/2								6	4			1									2	
M/3								5	4													
Y/1			1	1	2							1	1									
Y/2	2				3						1					1				2		2
Y/3	1			4	2							1				1		4		5		1
S/1		1			1						1	1										
S/2				1	2															2		
S/3	1	1	1				1					2			1		1		3		1	
D/1	2					1		1		2		1						1				2
D/2		1	1		2	1					6				1			2		2		
D/3	4			1	2						3	2	1					3		2	4	1
F/1	1			1	1			1	1					3	1							1
F/2													1	4	3				1			5
F/3	1													3	1							2
L/1							4						1									
L/2	2			1			2										7			2		

L/3	1				1		2				1		1		1		5				1	1

TABLE I-4 (continued). Number of Inconsistencies by Type and Question for Each Version.

T/1	1		1	1			1	4	1			1				3			1	1		
T/2	1			1				3		1		2							7		1	
T/3			1	1	1	1		4	1	1		3				2			5	1	3	

Each cell represents the number of inconsistencies found by subjects in that version under that question.

Key of Inconsistencies Found:

Physical Location - P Memory - M Virtual Device - D Semantic - S
Unavailable Task - T Syntactic - Y System Status (Feedback) - F Lexical - L

TABLE I-5. Seeded Number of Inconsistencies by Type and Question for Each Version.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
P/1					1																	
P/2	1	8	2	1	1	1				1		1			1					1		
P/3	2	5	2	3	2					1		1			1					1		
M/1							1															
M/2							2	1	1													
M/3							2	1	1													
Y/1	1				1																	
Y/2	3	1	1		4	1					1											
Y/3	2		1	2	4						1											
S/1				2	2			1														
S/2				1	2			1						1			2	2				1
S/3		1		1	2			1						1			1	2				
D/1																						
D/2											1							1		2	1	
D/3											1							1		1	1	1
F/1																	1					
F/2						1								1	3	1	1		1			
F/3		3				1								1	3	1	1		1	1		
L/1																						

L/2	3															2					
L/3	2															1					

TABLE I-5 (continued). Seeded Number of Inconsistencies by Type and Question for Each Version.

T/1			1					1													
T/2			1					1													
T/3			1					1													
Tot/ 1			1	2	4		1	2								1					
Tot/ 2	7	9	4	2	7	2	2	3	1	1	2	1		2	4	1	5	3	1	3	1	1
Tot/ 3	6	9	4	6	8	2	2	3	1	1	2	1		2	4	1	3	3	1	3	1	1

Each cell represents the number of inconsistencies found by subjects in that version under that question.

Key of Inconsistencies Found:

Physical Location - P	Memory - M	Virtual Device - D
Semantic - S		
Unavailable Task - T	Syntactic - Y	System
Status (Feedback) - F	Lexical - L	