Testing the Boundaries of Two User-Centered Design Principles: Metaphors and Memory Load

Misha Walker Vaughan
Department of Telecommunications
Indiana University

This experiment examined how varying implementations of two user-centered design principles affected the usability of a computer program. Drawing from human-computer interaction, as well as cognitive psychology, the following principles were implemented and tested: maximizing use of the user’s expectations and stereotypes and minimizing the user’s memory load. Nine interfaces were created using Hypercard’s datebook program. On each interface, the graphics and text were manipulated according to the design principle and the related cognitive psychology research. A total of 108 people participated. Of the five hypotheses, only one was supported by the data. The data suggest that for less complicated programs and tasks, strict adherence to design principles may not be required.

1. INTRODUCTION

Trade-offs in interface design can produce interfaces that deviate strongly from good user-centered design (UCD), or more severely, clearly violate UCD principles. For example, one common UCD principle suggests designing an interface using metaphors familiar to the user (e.g., Apple Computer, 1987). In addition, cognitive psychology suggests that simple metaphors are better (i.e., more usable) than complex metaphors (Manktelow & Jones, 1987). However, interface designers may find themselves needing to make trade-offs to meet real-world design demands. Such a situation might require choosing between (a) not using a metaphor at all, which would violate the design principle altogether and probably significantly

---

This research was supported in part by a grant from the Research and University Graduate School, Indiana University. A previous version of this article was presented at the International Communication Association annual conference in Chicago, May 23–27, 1996.

Special thanks are deserved by Walter Gantz for his unwavering support, demands of rigor, and attention to detail. Joe Hinshaw and Jeremy Hagemeyer also earned thanks for their contribution in data collection.

Misha W. Vaughan is now at the School of Library and Information Science, Indiana University.

Requests for reprints should be sent to Misha W. Vaughan, School of Library and Information Science, Indiana University, 4313 East Third Street, Bloomington, IN 47401. E-mail: milwalke@indiana.edu.
affect usability; or (b) using a more complex rather than a simple metaphor, which
would still be questionable according to cognitive psychology. A skilled designer
would most likely find a way to include the more complex metaphor rather than
not using a metaphor at all. However, a concern for the designer is whether this
trade-off, adhering to the spirit of the design principle but choosing the less optimal
implementation, will have an effect on usability.

This study sought to test this concern with two UCD principles: (a) maximizing
use of the user’s expectations and stereotypes and (b) minimizing the user’s memory
load. In particular, this study asked whether an interface design that adheres to the
spirit of these two design principles but falls outside the bounds of what cognitive
psychology would define as good design is significantly less usable. Focusing on
these two principles, a set of interfaces was implemented to test the proposition.
Because the intent was to compare a good design against a worse design, but not
against an unusable design (i.e., one that clearly violated the principle), the set of
interfaces ranged from good to poor as suggested by the cognitive psychology
literature.

To arrive at these two principles, cognitive psychology research (Gardiner &
Christie, 1987) was mapped onto a set of UCD design principles culled from the
human–computer interaction (HCI) literature (Apple Computer, 1987; Bastien &
Scapin, 1992; Brown, 1988; Cakir, 1986; Hooper, 1986; Hutchins, Hollan, & Norman,
psychology areas demonstrated strong overlap with the set of HCI design principles
and appeared to be the most straightforward to implement in a range from good to
poor: Manktelow and Jones’s (1987) discussion of mental models and Hitch’s (1987)
review of working memory. Mental models research clearly overlapped with the
HCI design principle of exploiting natural mappings, expectations, and stereotypes.
The literature on working memory proved to be strongly related to the HCI design
principle of minimizing short-term memory load. These two design principles
(exploiting natural mappings, expectations, and stereotypes and minimizing short-
term memory load) became a proving ground for determining the amount of
flexibility a designer has in adhering to UCD principles.

2. BACKGROUND

2.1. Exploiting Mappings, Expectations, and Stereotypes

Designing an interface to model a user’s real-world expectations and stereotypes is
one that builds on metaphors or analogies familiar to the user. Designing to mimic
the user’s natural or real-world mappings means the designer can rely on the user’s
expectations about those real-world referents to aid learning and use of the system.
However, this notion of real-world referents can be slightly misleading; different
people will have different real-world referents. A more useful approach to this
principle is to define it as design that mimics or draws from more familiar (rather
than less familiar) mappings, expectations, and stereotypes.
Also, there appear to be two levels on which to exploit mappings and stereotypes. One is conceptual and requires matching a computer function or operation with some real-world (or more familiar) equivalent, such as calling word processing “electronic typing.” The second level of mapping refers to matching the physical actions necessary to achieve a task to some real-world equivalent, such as using the right arrow key to indicate rightward motion. For this study, this design principle refers only to the first level, or conceptual level, of exploiting expectations and stereotypes.

The cognitive psychology literature provides an explanation for why this principle works as it does on both levels. According to Manktelow and Jones (1987), “Rational thinking depends on the construction of and operation of coherent and plausible mental models” (p. 108). Mental models are founded on a theoretical knowledge structure known as a schema. Schemata are ordered, higher order blocks of knowledge that contain what is typical of an experience, although not the actual record of the experience. Thus, a mental model draws from what is typical for the experience of the user. In turn, user-centered interface design should rely on typical experiences for the user, thus leading to the construction of good mental models.

Manktelow and Jones (1987) also suggested different designs that are ideal versus nonideal ways of achieving the development of mental models, the distinction between ideal and less than ideal being that the ideal design produces better human performance, which in this case means better usability. First, Manktelow and Jones argued that building a model based on the real world (or as is argued here, more familiar experience) is the ideal method and that building a model from given elements of the interface or system is less ideal. This leads to Hypothesis 1.

\[ H_1: \text{An interface designed to draw on more familiar metaphors is more usable than an interface that requires the user to develop a model from system metaphors.} \]

As a corollary to this hypothesis, Manktelow and Jones (1987) also argued that building a model from the user’s natural language (or more familiar language) is more effective than building a model from words defined by the system language.

\[ H_2: \text{An interface designed to draw on the user’s natural language will be more usable than an interface that uses system language.} \]

Second, Manktelow and Jones (1987) argued that building a model with complex metaphors makes it more difficult for the user than building a system based on simple metaphors.

\[ H_3: \text{An interface designed to draw on simple metaphors is more usable than an interface that draws on complex metaphors.} \]

2.2. Minimize Short-Term Memory Load

Minimizing a user’s short-term memory load as a design principle is more self-evident and suggests that designers should develop programs that minimize the
amount of work the user must do. The cognitive psychology literature offers an explanation for these short-term (or working) memory design suggestions (Hitch, 1987). The major conclusion of working memory research is the importance of capacity limitations in working memory, in that the user must keep track of multiple areas of information during any function. Those four areas of information are as follows: labels and parameters, current subgoals, current response, and current state of the computer. Thus, a design that interferes with any of these functions is interfering with working memory, not minimizing the user’s workload and reducing the program’s usability.

Two cases of ideal versus less than ideal implementation of minimizing short-term memory load can be drawn from Hitch (1987). First, Hitch argued that recognition of words and pictures is a more effective design than recalling the meaning of pictures.

\[ H_1: \text{An interface that draws on pictures and words is more usable than an interface that draws on only pictures.} \]

Hitch also argued that longer names for functions and tasks require more memory work and thus are less efficient than shorter names.

\[ H_2: \text{An interface that employs shorter names for tasks and functions is more usable than an interface that uses longer names.} \]

2.3. Usability

Nielsen (1993) and Shneiderman (1992) both provided good overviews of usability and differing approaches to measurement. For this study, usability was defined as the users’ efficiency, effectiveness, and satisfaction in using the programs. Efficiency was measured in terms of number of actions, effectiveness in terms of speed performance, and satisfaction in terms of the user’s subjective satisfaction. These three measures were most easily and reliably captured on the test program.

3. METHOD

3.1. Experimental Design

There were a total of nine interfaces designed for this program. One interface was the optimal interface; it possessed all of the positive characteristics in one design, that is, a more familiar metaphor, a simple metaphor, natural language, pictures and words, and shorter text. This was done because it would have been an impossible task to create different versions of an optimal design (along the previously mentioned lines) for each of the tests.

The maximizing expectations principle, as suggested by the hypotheses, was tested with three design elements: a system metaphor (vs. a more familiar meta-
phor), a complex metaphor (vs. a simple metaphor), and system language (vs. natural language). The system metaphor interface design and the complex metaphor interface design both had three versions: a graphics version, a text version, and a combined graphics and text version. The system language interface design used the textual version of the system metaphor interface. There were a total of six interfaces dealing with this principle.

The minimizing short-term memory load principle was tested with two design elements: pictures (vs. pictures and words) and longer text (vs. shorter text). The pictures interface only manipulated graphics, and the longer text interface only manipulated text. There were a total of two interfaces dealing with this principle.

3.2. Procedures

Six participants were recruited to pilot test the interfaces, the experimental setting, and the data collection method. The test site and experiment were modified on the basis of their responses.

A Macintosh 880AV computer was used to run the experiment and a video camera was used to record the participants’ interaction with the computer. The researcher led each participant through a short tutorial on the Macintosh interface and then explained the parameters of the experiment: The participant should work only with the displayed program and only with the keyboard (not the function keys) and mouse. The researcher handed each of seven tasks to the participant one at a time (Figure 1). The tasks had varying levels of difficulty, and their order was randomized to control for any task-order effect. In addition, the tasks were designed to be appropriate to the nature of the program.

The participants were randomly assigned to one of the nine interface conditions and to differing, predetermined task orders. One interface, the optimal design, was crucial to the experiment in that it would be the benchmark against which all the other designs would be compared. For this reason, this interface was oversampled in the experimental design. After the experiment, each participant completed a posttest satisfaction and demographic questionnaire.

3.3. Participants

One hundred twenty-one participants were recruited from three undergraduate classes: 52.1% were female, and 47.9% were male. The average age was 20.6 years.

Description: You have an appointment with Joe Marshak on April 29th, 1995 at 8:00 a.m.

Action: Enter "Joe Marshak—Breakfast" on April 29th, 1995 at 8 a.m.

FIGURE 1 A sample task given to participants.
and 89.3% of participants were White. Thirteen of the students failed to complete all of the tasks. Because a score on each task was necessary to compute the participant’s index scores, participants who did not complete a task were discarded, thus leaving 108 participants. All analyses were then run on this slightly smaller sample.

A distribution check was run on previous computing experience to confirm that each group for each interface had roughly equivalent previous computing experience. Table 1 demonstrates that the distributions for previous computing experience were similar for each interface group. No interface group was exceedingly high or low on prior experience, and all of the groups tended to clump between a score of approximately 20 and 30.

### 3.4. Treatment: The Interface Designs

A Hypercard datebook book program was used to provide the testable interfaces. The datebook program is distributed as part of the Apple Hypercard package. Apple Computer’s (1987) UCD philosophy is carried out in the datebook book program and thus provides an argument against the rival hypothesis that any negative effects on usability are due simply to a poorly designed program in general. Alterations to this program’s interface did not affect its functionality; each implementation of the interface could perform the same tasks. For this experiment, the choice was made to limit the interface designs to black and white and to manipulate only text and graphics. If color, sound, or animation had been introduced, it would have made the experimental design too complex for this project.

The different implementations of each interface were given to an interface designer for his evaluation of whether the actual design met the design condition’s objective. The interfaces were then altered according to his suggestions and comments. The interfaces went through this evaluation process twice.

First, the original interface was altered. In its original state (Figure 2), the interface contained background graphics; function buttons, including calendar, address, home, info, and so on; button graphics; a program name; and textual elements.

<table>
<thead>
<tr>
<th>Interface</th>
<th>M</th>
<th>Mdn</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28.2</td>
<td>28.5</td>
<td>28.0</td>
</tr>
<tr>
<td>2</td>
<td>29.0</td>
<td>28.0</td>
<td>27.0</td>
</tr>
<tr>
<td>3</td>
<td>26.2</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>4</td>
<td>25.2</td>
<td>26.0</td>
<td>26.0</td>
</tr>
<tr>
<td>5</td>
<td>27.9</td>
<td>28.0</td>
<td>16.0</td>
</tr>
<tr>
<td>6</td>
<td>23.0</td>
<td>20.0</td>
<td>17.0</td>
</tr>
<tr>
<td>7</td>
<td>27.0</td>
<td>25.5</td>
<td>21.0</td>
</tr>
<tr>
<td>8</td>
<td>27.6</td>
<td>28.0</td>
<td>20.0</td>
</tr>
<tr>
<td>9</td>
<td>28.1</td>
<td>30.0</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Note. The minimum possible score was 0.0 and the maximum was 67.0.
Certain elements were removed from the original interface, specifically the calendar and the address/info/home features. They were removed for two reasons: (a) to eliminate multiple paths for completing the same task, thus ensuring that each participant would have to perform the same minimum number of steps to complete each task, and (b) to accommodate the short time requirements by reducing the complexity of the program.

As stated previously, the first interface was designed to reflect the collective optimal recommendations of each hypothesis. This produced the optimal interface version (Figure 3). Specifically with regard to H1 (designing with a familiar metaphor), the datebook metaphor was the chosen familiar metaphor for the optimal interface. Graphically, the datebook metaphor comes through via the background graphic of a datebook with a spiral bind and with lines for each 30-min segment of the day. Also, the button graphics were altered to symbolize the features of the datebook: going to any day, going to today, and finding text on any day. (Whether these features were a good choice for Apple to incorporate into the original program’s design was not at issue; they were simply used for this experiment.) For the go to today button, rows and columns of boxes reminiscent of a calendar were used, but without indicating any particular day of the week. One box in that calendar is specifically highlighted, suggesting that one day, today, could be found by pressing the button. For the go to any day button, the same rows and columns visual was used; however, this time the image was overlaid by a question mark suggesting that the
user could find a day by clicking on this button. The find words button graphic used generic text in quotes.

In keeping with H_2 (designing with a simple metaphor), the datebook metaphor was left as it was; it was deemed simple enough. For H_3 (designing with natural language) and for H_4 (designing with words and pictures), text was added to each of the buttons indicating their function, that is, go to today, go to any day, and find words. Also for H_2, the name of the program itself “Datebook 1.3” was written across the top of the program window. Finally, for H_5 (designing with shorter text), all of the text on the interface was reread to ensure it was in as concise of a form as possible.

The rest of the interface designs worked off of this optimal model, changing it to fit each hypothesis. The system metaphor versions of the interface used the system metaphor of Hypercard, that of cards and stacks. The graphical version of the interface implemented this card and stack metaphor through the background graphic and through the button graphics. The background graphic was changed to reflect a stack of staggered cards. The button graphics were also changed to reflect a stack of staggered cards and finding information on these cards. A text version of this interface was also created. The textual element of this card and stack metaphor was conveyed through the name of the program by drawing on a system name, “Time Scheduler 1.3” (a program that simply helps the user schedule his or her time), and through changing the button labels to system labels. Each button name was changed to “Locate X” to reflect a programmer’s vision of the program as
something that locates information on cards, that is, locate today, locate any day, and locate words. The combined version used both the graphical and textual manipulations simultaneously.

The complex metaphor versions of the interface added to the datebook metaphor by adding a secretary agent between the user and the datebook. Metaphorically, the user asks the secretary to act on the datebook, rather than simply acting on the datebook himself or herself. This level of complexity was added only in terms of the graphical and textual elements; it did not actually add complexity to the functionality of the program itself. The interface manipulated graphically had a secretary graphic added to each button. The agent was designed to be both gender and racially neutral, somewhat like a gumball. The textually manipulated version changed the name of the program name to “Personal Secretary 1.3” and the button labels to turn to today, turn to any day, and find words. Also the words “Ask secretary to ...” were added to the interface to reinforce the notion that the user was posing requests to the secretary. The last part of H3 required combining both the text and graphic manipulations into an additional interface.

To test H4, whether an interface that used only pictures would prove less usable, the button labels from the optimal interface were removed, leaving only the button graphics.

H5 stated that an interface designed with longer text rather than shorter text would be less usable. To test this premise, another interface with longer text was created. The program name was lengthened to “Date Book Program 5.3,” and the button labels were lengthened to go to today’s page, go to any day’s page, and find words on a page.

3.5. Measures

The number of actions was measured by counting the total number of actions by a user for each task and then subtracting the minimum number of actions required to complete each task producing an index score. Actions were counted using the videotape of the participant’s performance. An action was defined as any discrete activity performed by the user, including performing a predetermined step, performing a step involved in committing an error, or performing a step involved in recovering from an error. Typing a name was considered one complete activity, but deleting a character and retyping it were counted as two distinct activities. The counting of actions began after the researcher stated the words “you may begin” on the videotape. Counting ended when the participant completed the last step required by the task.

Speed performance was also measured using the videotaped performance. Using a frame-accurate VCR, timing began immediately after the researcher stated the words “you may begin” and stopped when the participant performed the last step required of the task. This measure was also summed across each of the tasks, producing an index score for each participant.

The self-assessment measure of satisfaction was collected via a questionnaire. Using a 5-point Likert scale, the first eight items asked the participant to report on his or her personal experience of the program’s usability. The next set of 12 questions
asked about previous computing experience. The last four questions on the questionnaire asked for demographic information: age, race, year in school, and gender. The satisfaction items were summed, as were the previous computing items, to produce two more index scores for each participant.

3.6. Scales

Cronbach’s alpha for each of the scales was: previous computing experience \( \alpha = 0.65 \), self-report of the program’s usability \( \alpha = 0.86 \), speed performance \( \alpha = 0.68 \), and number of actions \( \alpha = 0.54 \). Several attempts were made to improve the alphas of previous computing experience, speed performance, and number of actions. A factor analysis revealed five factors in previous computing experience, two factors in speed performance, and three factors in number of actions. This suggested that each of these scales was multidimensional. An additional set of alphas was run to determine if the factors themselves produced more reliable indexes. They did not. Also, analysis of each set of factors revealed no discernible conceptual argument for their groupings. Ultimately, the decision was made to leave the indexes intact despite their relatively weak levels of internal reliability.

The indexes that required coding, speed performance, and number of actions were also checked for intercoder agreement. Ten percent of the videotaped participants were randomly selected and coded by a second researcher. Agreement on speed performance was defined as being within \( \pm 2 \) sec of the original time. Agreement on number of actions was defined as being within \( \pm 1 \) action of the original count. Agreement on the speed performance items overall was 91%. Agreement on the number of actions items overall was 72%.

4. RESULTS

4.1. Previous Computing Experience

A correlation matrix was run on the four scales to test whether previous computing experience was significantly related to any of the measures. A high degree of previous computing experience was significantly related to a positive self-report of satisfaction \( (r = .22, n = 108, p < .05) \) and a fast score on speed performance \( (r = -.50, N = 108, p < .01) \). These correlations suggested that the tests for the hypothesized differences would need to be run controlling for previous computing experience. To control for any variation due to previous computing experience, analyses of covariance (ANCOVAs) were run with previous computing experience as a covariate on \( H_1 \) and \( H_2 \). Because \( t \) tests were used for \( H_2, H_4, \) and \( H_5 \), previous computing experience participants were first divided into high- and low-computing-experience groups.

4.2. Learning Effects

A post hoc test on task order was run to check for any learning effects. The first task, regardless of which task it was, required approximately twice as much time and
twice as many actions than when it occurred elsewhere in the sequence. To control for any effect caused by learning time, the ANCOVAs were run controlling for task order (learning effect) as a second covariate. Dividing the t tests into yet two more subgroups would have reduced the power too much and thus were left as they were.

4.3. Hypotheses 1 and 2

H$_1$ stated that an interface that drew on a more familiar metaphor would be more usable than an interface designed to draw on a system metaphor. For this hypothesis, an ANCOVA was run between the relevant four interfaces while controlling for previous computing experience and learning (Table 2). These ANCOVAs were run on each of the usability measures: speed performance, number of actions, and self-report of satisfaction.

H$_2$ predicted that an interface in which the text was written in more familiar language would be more usable than an interface in which the text was written with system language. For this hypothesis, a t test was run on each index comparing familiar versus system text while controlling for previous computing experience (Table 2).

The data provided little support for H$_1$ or H$_2$. The p values for the main effects and the two-way interactions on each index were nonsignificant. The t test values on each index also were not significant. However, the means for the speed index, the actions index, and the self-report index did fall in the predicted direction. This can be seen by comparing the theoretically worst interface (bottom right-hand cell for each index) with the theoretically best interface (top left-hand cell for each index). The speed performance means for best and worst interface were 490.10 and 581.65, respectively (lower mean = better usability). The means for number of actions for best and worst interface were 44.65 and 60.45, respectively (lower mean = better usability), and the means for the self-report index were 34.60 and 33.00 for best and worst interface (higher mean = better usability). With H$_2$, the means for two of the t tests also were in the predicted direction. On speed performance, the means were 490.10 and 494.21 for best and worst interface, respectively. On the self-report index, the means were 34.60 and 32.32 for best and worst interface design, respectively. Although these data did not support H$_1$ and H$_2$, they were not antithetical to the premises associated with these hypotheses.

4.4. Hypothesis 3

H$_3$ stated that designs that draw on simpler metaphors would be more usable than those that rely on more complex metaphors. To test this hypothesis, an ANCOVA was run between four interfaces on speed performance, number of actions, and self-report of satisfaction while controlling for previous computing experience and learning (Table 3).

There were no significant differences among the four interfaces on any of the indexes assessed. The p values for the main effects and the two-way interactions
Table 2: Analysis of Covariance, t Test Means for Hypothesis 1 and Hypothesis 2: Speed Performance, Number of Actions, and Satisfaction on the Familiar Metaphor Interface Versus the System Metaphor Interface Controlling for Previous Computing and Task Order

| Graphics | Speed Performance | | Number of Actions | | Self-Report of Satisfaction |
|----------|-------------------|---------------|-------------------|-------------------|
| Familiar | Graphics          | Score | n | Score | n | Score | n |
| System   |                   | 490.10^a    | 20 | 514.40 | 10 | 34.60^a | 20 | 34.00 | 10 |
| Familiar |                   | 494.21^a    | 14 | 581.64 | 11 | 32.21^a | 14 | 33.00 | 11 |

The lower the mean the more usable the interface; N = 55; Covariate: previous computer experience (df = 1; F = 22.662; p < .000); Covariate: task order (df = 1; F = 3.286; p < .076). *Main effects: graphics (df = 1; F = .400; p < .530). *2-Way: Graphics x Text (df = 1; F = .304; p < .584). ^Main effects: text (df = 1; F = .194; p < .661). ^t = -.36, p < .720. The lower the mean the more usable the interface; N = 55; Covariate: previous computer experience (df = 1; F = 4.438; p < .040); Covariate: task order (df = 1; F = 2.270; p < .138). *Main effects: graphics (df = 1; F = .001; p < .982). *2-Way: Graphics x Text (df = 1; F = 1.231; p < .273). ^Main effects: text (df = 1; F = 2.751; p < .104). ^t = -.60, p < .554. The higher the mean the more usable the interface; N = 55; Covariate: previous computer experience (df = 1; F = 4.471; p < .040); Covariate: task order (df = 1; F = .319; p < .575). *Main effects: graphics (df = 1; F = 1.661; p < .204). *2-Way: Graphics x Text (df = 1; F = .392; p < .534). ^Main effects: text (df = 1; F = .251; p < .614). ^t = .41, p < .687.
Table 3: Analysis of Covariance Means for Hypothesis 3: Speed Performance, Number of Actions, and Satisfaction on the Simple Metaphor Interface Versus the Complex Metaphor Interface Controlling for Previous Computing and Task Order

<table>
<thead>
<tr>
<th>Graphics(^{bc})</th>
<th>Speed Performance(^3)</th>
<th>Number of Actions(^5)</th>
<th>Self-Report of Satisfaction(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Simple})</td>
<td>(\text{Complex})</td>
<td>(\text{Simple})</td>
<td>(\text{Complex})</td>
</tr>
<tr>
<td>Score</td>
<td>(n)</td>
<td>Score</td>
<td>(n)</td>
</tr>
<tr>
<td>490.10</td>
<td>20</td>
<td>600.43</td>
<td>7</td>
</tr>
<tr>
<td>481.33</td>
<td>9</td>
<td>522.33</td>
<td>12</td>
</tr>
</tbody>
</table>

\(^3\)The lower the mean the more usable the interface; \(N = 48\); Covariate: previous computer experience (\(df = 1; F = 10.514; p < .002\)); Covariate: task order (\(df = 1; F = .049; p < .825\)).
\(^5\)Main effects: graphics (\(df = 1; F = .266; p < .608\)).
\(^6\)Main effects: Text (\(df = 1; F = .965; p < .331\)).
\(^7\)The higher the mean the more usable the interface; \(N = 48\); Covariate: previous computer experience (\(df = 1; F = .000; p < .984\)); Covariate: task order (\(df = 1; F = .343; p < .561\)).
\(^8\)Main effects: graphics (\(df = 1; F = .256; p < .616\)).
\(^9\)Main effects: text (\(df = 1; F = .466; p < .498\)).
were not significant. However, a comparison of means on the most usable and least usable interfaces on each index showed the predicted pattern for the data. On speed performance, the theoretically most usable interface had a better speed performance score (490.10 sec) than did the theoretically worst interface (522.33 sec). On number of actions, the most usable interface had a better score (44.65) than did the worst interface (54.75). Indeed, the most usable interface had the best score in this regard. Finally, the best interface also had a better self-report score (34.60) than did the worst interface (32.83). Despite this pattern of scores, the lack of significant difference requires that H₃ be rejected.

4.5. Hypothesis 4

H₄ stated that an interface designed with both pictures and words would be more usable than an interface designed with only pictures. To test this, a t test was run between two interfaces, one with both pictures and words and one with only pictures. To control for previous computing experience, the sample was split into two groups based on previous computing experience (low and high); a median split was employed. T tests were then run on each of the three indexes, speed performance, number of actions, and self-report (Table 4).

Three statistically significant differences emerged. The number of actions was significant for both groups of users; the pictures and words interface was more usable than the pictures only interface. Less experienced users averaged 47.30 actions for the pictures and words interface and 72.71 for the pictures only interface. More experienced users averaged 42.00 actions on the pictures and words interface compared with the pictures only interface mean of 68.17. The self-report of satisfaction was also significant for less experienced users; the pictures and words interface earned a mean of 34.60, whereas the pictures only interface earned a mean of 29.57.

Even though not significant, speed performance for both groups and satisfaction for more experienced users reflected the same trend, with the pictures and words interface rating more usable than the pictures only interface. Overall, H₄ received partial support. These data suggest that a more usable interface would employ both pictures and words rather than pictures alone.

4.6. Hypothesis 5

H₅ stated that an interface designed with shorter text would be more usable than an interface designed with longer text. T tests on each index were run between the shorter names interface and the longer names interface (Table 5). Previous computing experience was controlled for in the same way as the previous t tests.

The variation in text length did not significantly influence responses on any of the usability indexes. However, examination of the means on each of the indexes showed some support for the predicted pattern of usability. For less experienced users, the shorter text interface resulted in fewer actions (47.30) than the longer text interface (55.60). However, the reverse was true for more experienced users. They
had more actions on the shorter text interface (42.00) than the longer text interface (29.29). The same pattern was reflected in the self-report data. Less experienced users evaluated the shorter text interface as more usable (34.60) than the longer text interface (33.00). More experienced users evaluated the longer text interface as more usable (34.60–37.29). Collectively, these data provide no support for H₆.

5. DISCUSSION

Each of the hypotheses argued that strict adherence to UCD principles (as defined by cognitive psychology research) would produce more usable interface designs. However, only one received support—H₅—and support for that hypothesis is
eclipsed by the complete lack of support on the other four. To explore whether differences in usability might be occurring when a user was initially learning a program, a post hoc test was conducted. A final series of ANCOVAs and t tests were run, using only the first task (in which learning seemed to be occurring) for each participant. No significant differences were found across conditions. Thus, the possibility that usability may be particularly affected during the initial contact with this program was not supported.

The two interface design principles seem to have a higher degree of latitude in implementation than is suggested by cognitive psychology. Collectively, the data argue that these two interface design principles can be implemented within a certain degree of variability that does not significantly affect usability. For designers, this suggests that there is room within a program’s “usability window” to accommodate different design trade-offs, as long as the trade-offs still follow the spirit of a design principle.

However, there are some qualifications of this conclusion. First, it is possible that the tasks themselves were not sufficiently complex or of sufficient duration to capture the true impact of design differences. Second, the program used to test these hypotheses was pared down to four functions (enter an event, go to a day, find a day, and search for text). It is possible that differences were not identified because of the program’s simplicity. Third, it has been suggested that the interface implementations may not have been significantly different. This critique misses the nature of this study, which was to test whether interface designs driven by variability by cognitive psychology research and yet bounded by adherence to a design principle, would produce significantly different usability. The answer in this case was no. Finally, the participants used in the experiment all had some computing experience. It is possible that a broader cross-section of the population would have added people who are more intimidated by computers or people with less computer experience. Either factor might predispose a person to be more sensitive to interface design, thus making usability more significant.

These qualifications deal with weaknesses in the experimental design. However, there are two more alternative explanations for the lack of significant findings. Both alternatives suggest a need for improving our theoretical model and, concomitantly, our understanding of HCI.

First, it is possible that cognitive psychology overestimates the power of previously constructed stereotypes, expectations, and maps, as well as the need to minimize short-term memory load. Instead of relying on existing mental models to guide their understanding and use of a computer, users may be considerably more attentive, open, and willing to learn the models inherent to a computer program. Also, users may possess a better ability for maintaining information about computer programs in working memory than is currently suggested.

A second theoretical explanation, or correction, stems from underestimating the power of the human mind. It is possible that in building the theory surrounding HCI, researchers have underestimated the human capacity to adapt to new situations and phenomena with regard to computers. If this were true, then interface design principles and guidelines can rightly argue for a range or flexibility in
implementation. This would also explain why researchers might have erred in theorizing about the these two design principles.

Clearly, a future study ought to focus on more complex designs of the interfaces and tasks. A more complex program and a longer, more complex task design would allow for a more robust test of the hypotheses. In addition, a future study should seek to add a wider range of computing backgrounds to the participant group, as well as test for computer anxiety before the experiment.

REFERENCES


