A Comparison of Static, Adaptive, and Adaptable Menus

Leah Findlater and Joanna McGrenere
Dept. of Computer Science, University of British Columbia
Vancouver, B.C., Canada
{lkf, joanna}@cs.ubc.ca

Abstract
Software applications continue to grow in terms of the number of features they offer, making personalization increasingly important. Research has shown that most users prefer the control afforded by an adaptable approach to personalization rather than a system-controlled adaptive approach. No study, however, has compared the efficiency of the two approaches. In a controlled lab study with 27 subjects we compared the measured and perceived efficiency of three menu conditions: static, adaptable and adaptive. Each was implemented as a split menu, in which the top four items remained static, were adaptable by the subject, or adapted according to the subject's frequently and recently used items. The static menu was found to be significantly faster than the adaptive menu, and the adaptable menu was found to be significantly faster than the adaptive menu under certain conditions. The majority of users preferred the adaptable menu overall. Implications for interface design are discussed.

Categories & Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – evaluation/methodology, interaction styles.

General Terms: Design, Experimentation, Human Factors, Performance.

Keywords: Adaptive interfaces, adaptable interfaces, customization, menu design, user study, interaction techniques.

INTRODUCTION
Everyday applications, such as the word processor and the spreadsheet, provide additional functionality with each new version release. Some have referred to this phenomenon as creeping featurism [4, 13] or bloatware [6]. One impact of this trend is that graphical user interfaces are increasing in complexity – menus, toolbars, and dialog boxes are all multiplying in size. On the positive side, the addition of new features can provide benefit to the user; for example, a feature may modernize an application, as in the case of a word processor that adds support for creating an html document for web publishing. The downside, however, is that most users only use a small fraction of the available functions [8, 11], and must, therefore, wade through many unused functions. More so than ever before, there is a need to manage the interface, providing users with easy access to the functions that they do use. In addition, users tend to use different functions from one another, even when they are performing similar tasks [1]. This suggests the need for interfaces to be personalized to each individual user.

There are two main approaches to personalization. Adaptive interfaces dynamically adjust the interface in a way that is intended to support the user. By contrast, adaptable interfaces provide customization mechanisms but rely on the user to use those mechanisms to do the adaptation. These approaches differ with respect to who is in control of the personalization: adaptive interfaces are system-controlled whereas adaptable interfaces are user-controlled. Traditionally the system designer or administrator has also played a role in adapting the interface to the needs of a particular user or group; however, adaptable and adaptive interaction techniques are likely the only scalable approaches to personalization [18].

There has been some debate in the HCI community as to which of these two approaches is best [15]. One side argues that we should provide easy-to-use predictable mechanisms that keep users in control of their system, while the other side believes that if the right adaptive algorithm can be found, users will be able to focus on their tasks, rather than on managing their tools. Despite this debate, there has never been an empirical comparison of the efficiency of adaptive and adaptable interaction techniques. Our work provides a first step towards addressing that gap.

We document an experiment with 27 subjects in which we compared the performance (speed and error rate) of three menu designs: static, adaptive, and adaptable. Each was implemented as a split menu, in which the top four menu items remained static, were adaptable by the subject, or adapted according to the subject's most frequently and recently used items. The static menu was found to be significantly faster than the adaptive menu. In addition, under certain conditions, the adaptable menu was found to be significantly faster than the adaptive menu but not significantly different from the static menu. The majority of subjects preferred the adaptable menu.

CHI 2004, April 24–29, 2004, Vienna, Austria.

TERMINOLOGY

To provide consistent terminology throughout the paper, we present general definitions on interface variability. Variability refers to whether or not an interface changes over time, and can have one of two values:

Static: The interface does not change during the course of use.

Dynamic: The interface changes during the course of use.

For dynamic interfaces, there are three possibilities for controlling changes to the interface:

Adaptive: The system controls change.

Adaptable: The user controls change. (Another term for this is customizable.)

Mixed-initiative: The control is shared between the user and the system.

RELATED WORK

Our work extends the design of split menus, introduced by Sears and Shneiderman [14]. Split menus contain two partitions in order to facilitate faster access to frequently used menu items. Those items that are accessed frequently are located in the top partition, above "the split". In both a controlled experiment and a field study, Sears and Shneiderman showed that a static split menu, which contained predetermined frequent menu items in the top partition and did not change during the course of use, was at least as fast as a static traditional menu, and in most cases significantly faster. In our experiment, we extend their split menu design to include an adaptable variant, where the user can specify the items in the top partition. Their work suggested the need for an adaptive split menu, where the items in the top partition dynamically adapt based on a user's usage pattern, but they did not evaluate such a design in their studies.

To date, adaptive interfaces have appeared mostly in the research literature, rather than in industrial products, and have not often been compared to non-adaptive interfaces. A few exceptions are noted here. One example from industry is the menus in the Microsoft Office 2000 suite, which were significantly redesigned from those in Microsoft Office 97 [12]. These are dynamic menus, which adapt to an individual user's usage. When a menu is initially opened, a "short" menu containing only a subset of the menu contents is displayed by default. To access the "long" menu one must hover in the menu with the mouse for a few seconds or click on the arrow icon at the bottom of the short menu. When an item is selected from the long menu, it will then appear in the short menu the next time the menu is invoked. After some period of non-use, menu items will disappear from the short menu but will always be available in the long menu. Users cannot view or change the underlying user model maintained by the system; their only control is to turn the adaptive menus on/off and to reset the data collected in the user model.

Greenberg and Witten compared the performance of an adaptive menu to a static menu [2]. In a controlled experiment, users were asked to search for names in a telephone directory in each of the two menu conditions. Results showed that the adaptive structure, which provided a shorter search path to the most frequently accessed items, was faster than the static structure. No adaptable condition was evaluated.

One comparison of adaptive and adaptable interfaces was a longitudinal field study done by McGrenere et al. in which a prototype adaptable interface for Microsoft Word 2000 (MS Word 2000) was compared to the native adaptive interface described above [10]. Their adaptable interface included two interfaces between which the user could easily toggle: a dynamic personalized interface that the user constructed to include only desired functions, and the static full interface of MS Word 2000. The study showed that, given an easy-to-use customization facility, the majority of participants were able to customize their personal interfaces according to the functions they used. Most participants also preferred the adaptable interface to the native adaptive interface. The study, however, did not compare the performance of these approaches. Our study extends this research by specifically addressing the efficiency of the menu designs. The combination of results from the longitudinal field study and our controlled lab experiment provides a more complete understanding of the adaptable versus adaptive debate than either study provides on its own.

Cognitive modeling has also been used to study adaptive systems. Warren uses the Model Human Processor to predict the cost or benefit of using an "intelligent" or adaptive split menu over a static split menu in a diagnosis system for physicians [17]. Results from applying the model show that the adaptive system was beneficial in theory; however, their model assumed that the user does not have enough familiarity with the menu to anticipate item locations. No adaptable design was evaluated.

There has been relatively little research on customization. In the context of the UNIX operating system, Mackay found that actual customization is minimal because of time, difficulty, and lack of interest [9]. One way to interpret Mackay's work is that customization facilities need to be easy to use if we are to expect users to customize. Although the result may seem obvious, it is not clear that this has been recognized by industry. By showing that users can customize effectively, we hope to motivate the design of easy-to-use customization facilities.

There has also been a small amount of research into interfaces that combine both adaptive and adaptable elements, namely mixed-initiative interfaces [3]. One application example was introduced by Thomas and Krogsœter, who extended the user interface of a common spreadsheet application and showed that an adaptive component which suggests potentially beneficial adaptations to the user could motivate users to adapt their interface [16]. More recently, Jameson and Schwarzkopf studied the issue of controllability in an adaptive recommendation system for choosing conference itineraries [5]. Their results were inconclusive.

EXPERIMENTAL METHOD

The goal of our experiment was to compare the efficiencies of static, adaptable, and adaptive menus. Each menu was implemented as a split menu, a design briefly introduced in the previous section.

Items are placed in the top or bottom partition of the split menu based on the frequency with which each item has been selected in the past. This is not done in real-time, but is done when the menu is initially setup as a split menu. In a traditional static menu (one that does not contain a split), items may be ordered by strategies such as alphabetic or functional ordering. In a split menu, this relative ordering of items is maintained within each partition. For example, if Canada appears before Uganda in the traditional menu layout, it will appear before Uganda if both are in the top or bottom partition of the split menu.

Sears and Shneiderman used the following two preliminary design guidelines for their studies [14]:

1. At most four items should appear in the top partition.
2. To split the menu into two partitions, first sort items by frequency. Starting with the lowest frequency item, scan the list until the point when the difference between successive frequencies is greater than the mean of all frequencies. The top items on the high frequency side of that point are assigned to the top partition, up to a maximum of 4.

We adopted the first guideline, but relaxed the second one such that four items always appeared in the top partition for all three menu conditions. This was done so that the size of the top partition would not be a confounding factor.

Menu Conditions

Static split menu

This is a classic split menu. The items in the top partition are the four most frequently occurring items in the selection sequence of the experimental task. Thus, this menu represents the ideal static split menu for the task. Figure 1(a) shows an example of such a menu. Note that we did not explicitly include a traditional static menu specifically because previous results showed that a static split-menu is at least as efficient as a traditional static menu [14].

Adaptive split menu

An adaptive algorithm dynamically determines which items should appear in the top partition of the menu, based on the user's most frequently and recently used items. These are the two main characteristics of the Microsoft adaptive algorithm [12], and are also the two suggested by the literature [1]. Our goal was to create as predictable an algorithm as possible, based on these two characteristics.

Our adaptive partitioning algorithm is shown in Figure 2. It designates two items in the top partition to be *frequency items*, and two to be *recency items*, and always ensures that there are four items in the top partition. The initial items in the top partition of the adaptive menu are the same as those in the top partition of the static menu.

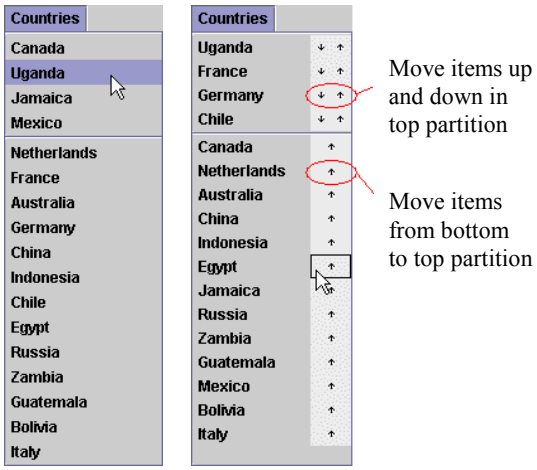

Figure 1. (a) Static split menu; (b) Adaptable split menu.

Adaptable split menu

The adaptable menu is a dynamic menu controlled by the user. An important goal of the adaptable menu design was to make the adaptation process as simple as possible. Two levels of customization are provided: coarse-grained customization allows items to be moved to the top or bottom partition; fine-grained customization allows items to be positioned in specific locations within the top partition. As shown in Figure 1(b), arrow buttons allow the user to perform this customization with a single click. Note that the fine-grained customization allows an extra degree of precision not available in the other two menu conditions. The order of items in the bottom partition, however, remains static.

The top partition of the adaptable split menu is initially left empty and it is the user's responsibility to add items (to a maximum of four). The reason for this is that the literature

```
(initially: item.frequency = item.recency = 0)

selectedItem.frequency++
selectedItem.recency = 0
for each remaining item[i] in the menu
    item[i].recency++
    if the selected item is in the top partition already
        do nothing
    else
        leastRecItem =  least recently used of the recency items
        leastFreqItem = least frequent of the frequency items
        if leastRecItem.frequency < leastFreqItem.frequency
            move leastRecItem to bottom partition
        else
            move leastFreqItem to bottom partition
            set leastRecItem.type = frequency item
        move selectedItem to top partition of menu
        set selectedItem.type = recency item
```

Figure 2. Adaptive algorithm.

File	Insert	Format
12	74	2
9	0	25
1	0	7
5	0	15
295	0	0
1	2	2
0	6	0
1	2	6
16	0	19
68	0	0
130	42	0
0	0	0
1	0	18
2	0	19
0	8	0
	0	
	0	

Figure 3. Selection frequencies from original log data for each menu item location; e.g., the first item in the File menu was selected 12 times (*n*=788).

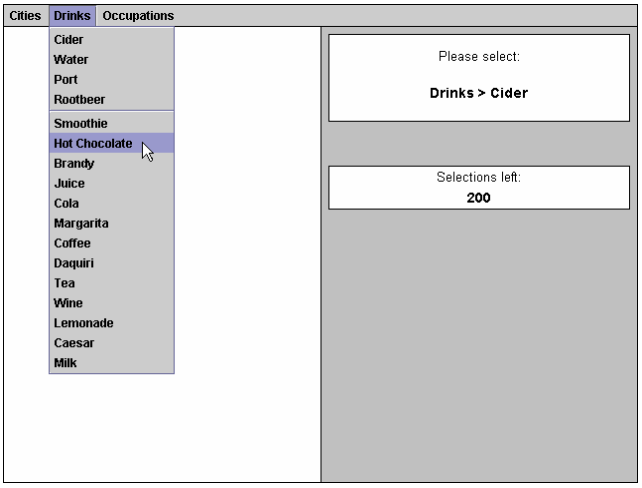

Figure 4. The experimental system, showing the prompt area on the right-hand side and the menus at the top-left.

suggests that users are reluctant to customize unless forced to do so [9]. By placing default items into the top partition, we expected subjects to be less willing to customize.

Task

We took a similar approach to task construction as Greenberg and Witten [2]. Namely, we simulated a real-world task by constructing the experimental task based on real menus and real menu selection data. We used 20 weeks of log data from an office administrative assistant's use of MS Word 2000[1]. The original 11,000 entries in the log included toolbar and shortcut key selections as well as menu selections. After extracting all but the menu selections, 1387 items remained.

Basing the task on the MS Word menu structure and usage data allowed us to assess the efficiency of the three menu conditions on realistic menu lengths and complexities. While the use of a real task scenario would have given subjects context to help them understand the benefit of adapting their menus, this could have confounded performance results. For example, if the task were to format a document in MS Word, subjects' existing familiarity with the software and ability to format documents could have accounted for most of the variability in performance. For this reason, we abstracted away the task scenario, and set the task as a sequence of menu selections. This is similar to Sears and Shneiderman's second experiment [14].

We selected the three most frequently accessed MS Word menus from the log data: File, Format and Insert, which together accounted for 788 selections. The selection rates for these menus are shown in Figure 3. To avoid transfer effects, we *masked* these menus using different labels. For example, the File->New menu item became the Cities->Paris menu item. The Cities menu contained 15 different city names, just as the File menu in MS Word contains 15 items. The items in the Cities menu were not ordered alphabetically, just as those in the File menu are not.

Given our three menu conditions, we required three isomorphic tasks (structurally equivalent tasks). This was achieved in two steps. (1) We permuted the order of the underlying MS Word menus as follows File/Format/Insert, Format/Insert/File, and Insert/File/Format respectively, so that menu selections from the underlying File menu would not always come from the leftmost menu in our experimental setup. (2) For each of the 3 permuted orders, a different mask was created for the {File, Insert, Format} triplet; for example, {Cities, Drinks, Occupations} was one of the masks. The result was three different *schemes*.

A task block consisted of a 200-item selection sequence. This corresponded to roughly a 4-week period from the original data. The sequence was a contiguous block of selections from the 788 File, Format, and Insert selections, thus reflecting temporal patterns in the original data. The starting index for each subject's sequence was chosen randomly between 1 and 588, to mitigate the influence of any specific 200-item sequence. The experimental system is shown in Figure 4.

Measures

We included both quantitative and qualitative measures. Our main dependent variable was speed. An implicit error penalty was included by forcing subjects to correctly select an item before continuing to the next menu selection. Error rate was also recorded independently for completeness.

A poll-style questionnaire was administered after all conditions had been completed. Subjects were asked to rank order each menu condition according to the following criteria: overall preference, efficiency, error rate, frustration, and initial ease of use.

[1] The log data from this administrative assistant was collected in the year 2000, with the adaptive menus in Word 2000 turned off. She had been using MS Word since 1987 and on average spent 3 to 5 hours per day word processing. She identified herself as an intermediate MS Word user.

Design
The design was a within-subjects 3x3 (menu condition x scheme) factorial. A within-subjects design was chosen for its increased power and because it allowed for comparative comments on the three menu conditions. To minimize learning effects we counterbalanced the order of presentation using a Latin square for menu condition and a Latin square for scheme, resulting in nine configurations.

Subjects
Subjects were undergraduate students recruited through the Department of Psychology and were paid $10 to participate for one hour. A total of 27 subjects were used, 3 subjects per configuration. All were between the ages of 18 and 39; there were 6 males and 21 females. Ten described themselves as novice computer users, 16 as intermediate users, and 1 as an expert user.

Motivation
To motivate subjects to perform the task quickly, subjects were told that an extra $10 would be provided to 1/3 of the subjects who completed the selection blocks the fastest. The 1/3 ratio was chosen to encourage subjects to believe they had a reasonable chance of being paid the additional $10.

Apparatus
The experiment was conducted on five Apple eMacs running Mac OS 10.1.4 with Power PC G4 processors and 128 MB RAM. The experimental system, including all menus, was fully automated and was coded in Java 1.3.1.

For each selection, a menu and item prompt was presented on the screen, for example, Drinks->Cider (as shown in Figure 4). When the subject selected this item, the prompt changed to the next item in the selection sequence. Errors were indicated by the addition of a red 'X' next to the prompt, and the subject had to correctly select the prompted item before continuing.

Procedure
The experiment was designed to fit in a single one hour session. The procedure was as follows. (1) A questionnaire was used to obtain information on user demographics and computer experience. (2) Subjects were given a practice session of 20 selections on a static split menu, followed by a 1 minute break. (3) For each of the three menu conditions, the system provided a one or two sentence description and indicated that two identical sequences of selections would be given with a short break in between. Block 1 consisted of a 200-item sequence and was intended to be a warm-up block. Block 2 consisted of the identical 200-item sequence to Block 1. Between the two blocks, subjects were given a 2 minute break. For the adaptable condition, subjects were allowed to take extra time during the break to customize their menus if they wished to do so. That was their only opportunity to customize. Each condition took approximately 12 minutes from start to finish, and there was a 5 minute break in between conditions. (4) A feedback questionnaire was used to rank the menu conditions on the qualitative dependent variables and to record additional comments. Brief, informal interviews were also conducted with some of the subjects based on their questionnaire data.

Instructions for customization were included in the experimental system after Block 1 of the adaptable condition as follows: "To customize the menus above, you can move items to the top section of the menu using the single up arrows. There can be at most four items in the top section."

Note that each subject received the identical 200-item sequence for all three menu conditions, although masked with a different scheme.

Hypotheses
Our hypotheses were as follows:

H1. Adaptive is slower than both static and adaptable.
H2. Adaptable is not slower than static.
H3. Adaptable is preferred to both adaptive and static.
H4. Static is preferred to adaptive.

RESULTS
Here we describe the experimental results, both from the measured performance data as well as from the self-reported questionnaire data. From our 3x3 (menu condition x scheme) design, there were no significant main, interaction, or ordering effects of scheme, so we examine only the effects of menu condition and its ordering.

Customization
Of the 27 subjects, only 22 customized based on the task sequence. Three subjects did not customize at all, and two did not appear to understand the customization process, each placing items in the top partition of only one out of the three menus before continuing to Block 2. For one of these subjects, none of the customized items were ever used in the selection sequence, and for the other, two of the items were never used. The average time spent on customization for all subjects was 142 seconds (N=27).

Performance
To determine if there were differences in performance between the menu types, two-way ANOVA's (Order x Menu) and post-hoc pair-wise comparisons were calculated for both speed and error dependent measures on the data recorded in Block 2. Along with statistical significance, we report partial eta-squared (η^2), a measure of effect size. Effect size is a measure of practical significance, and is often more appropriate than statistical significance in applied research in Human-Computer Interaction [7]. To interpret partial eta-squared, .01 is a small effect size, .06 is medium, and .14 is large.

Speed
The overall results for speed are shown graphically in Figure 5. There was a significant difference in speed based on the menu type used, i.e., a significant main effect of Menu ($F(1.44, 34.64) = 12.54$, $p < .001$, $\eta^2 = .343$). Additionally,

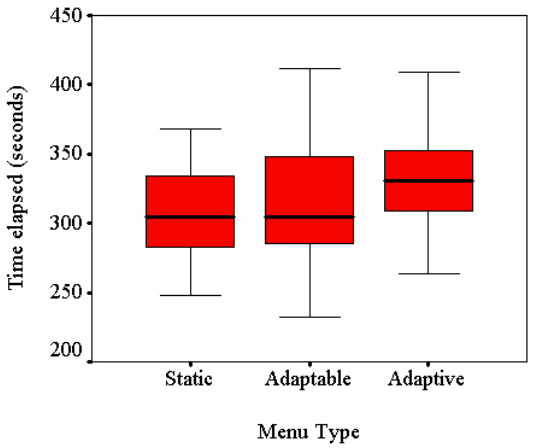

Figure 5. Boxplot of menu type versus speed (N=27); the line through each box represents the median.

the order that the menu conditions were presented disproportionately affected the speed of performance for each menu type, i.e., a significant interaction effect between Order and Menu ($F(2.89, 34.64) = 6.14$, $p = .002$, $\eta^2 = .338$). (The data was non-spherical, hence we used the Greenhouse-Geisser adjustment.)

We had expected to find an ordering effect, whereby subjects would perform more quickly as the experiment progressed, but we had not expected an interaction effect between Order and Menu[2].

To examine these effects in more detail, we compared each possible pair of menu types for each of the three orders. The results are shown in Table 1. This was done with post-hoc pair-wise comparisons, computed using a Bonferroni adjustment. The three orders were: (1) Static-Adaptive-Adaptable, (2) Adaptive-Adaptable-Static, and (3) Adaptable-Static-Adaptive. The comparisons showed that static was significantly faster than adaptive for all three orders ($p = .002$, $p < .001$, and $p < .001$ for Orders 1, 2, and 3 respectively).

The interaction effect can be explained by looking at the relationship between the adaptable menu condition and Order. When adaptable was *not* presented first, it was significantly faster than adaptive ($p = .041$ and $p = .026$ for Orders 1 and 2 respectively). In addition, it was not significantly different from the static condition. When adaptable was presented first (Order 3), however, adaptable was significantly slower than static ($p = .001$), and not significantly different from adaptive. In other words, those subjects who saw the adaptable condition first were significantly slower, relative to the other conditions, than those subjects who saw the adaptable condition as the second or third condition. In fact, Table 2 shows that the mean speed for the adaptable condition in Block 2 drops to 300.72 when only the 22 subjects who customized are considered. This is almost identical to the static condition (301.76). Four out of the 5 subjects who did not customize were in the adaptable condition

[2] Pilot testing with 8 subjects did not reveal this interaction effect.

Table 1. Pairwise comparisions for speed (N=27).

Menu (*i*)	Menu (*j*)	Mean Difference (*i-j*)	Std. Error	Sig.[a]	Partial η^2
Order 1: Static-Adaptive-Adaptable					
Static	Adaptable	6.084	9.785	1.00	.065
Static	Adaptive	-20.787*	5.356	.002	.681
Adaptable	Adaptive	-26.871*	10.082	.041	.556
Order 2: Adaptive-Adaptable-Static					
Static	Adaptable	-.623	9.785	1.00	<.001
Static	Adaptive	-29.478*	5.356	<.001	.719
Adaptable	Adaptive	-28.854*	10.082	.026	.500
Order 3: Adaptable-Static-Adaptive					
Static	Adaptable	-42.327*	9.785	.001	.672
Static	Adaptive	-25.048*	5.356	<.001	.810
Adaptable	Adaptive	17.278	10.082	.298	.225

a. Adjusted for multiple comparisons using Bonferroni.
* The mean difference is significant at the .05 level.

first (Order 3), which explains the interaction effect found. Implications of this finding are considered in the Discussion section.

Error rate

On average, 6.8, 6.6, and 6.9 errors were made during Block 2 of the static, adaptable, and adaptive menu conditions respectively. A two-way ANOVA showed a significant interaction effect between Order and Menu ($F(4,48) = 2.94$, $p = .030$). Using a Bonferroni adjustment, however, no pair-wise comparisons were significant.

Self-reported Measures

For each of the five self-reported variables, we analyzed the frequency with which each menu condition was ranked first. This was done by calculating the Chi-square statistic to determine if actual frequencies were significantly different from the case in which all frequencies are equal. A summary of the results is shown in Table 3[3]. There were no significant correlations between order and the results for any of these variables.

Chi-square was significant for four out of the five self-reported measures. For overall preference, the most popular choice was the adaptable menu (15 subjects, $\chi^2(2,27) =$

Table 2. Means for selection Speed of all subjects, and for only those subjects who customized (N=27).

	All 27 Subjects		22 Who Customized	
Menu	*Block 1*	*Block 2*	*Block 1*	*Block 2*
Static	327.23	306.51	323.79	301.76
Adaptable	410.71	318.80	402.28	300.72
Adaptive	354.22	331.62	352.37	326.86

[3] Due to improperly completed questionnaires, two data points were missing for Perceived Error Rate, and one datum was missing for each of Frustration and Initial Ease of Use.

Table 3. Chi-square statistic for qualitative results (S=static; AV=adaptive; AB=adaptable) (N=27).

Dependent Variable	Ranked 1st (fre- S	AV	AB	df	Chi-square	Sig. Level
Preferred overall	4	8	15	2	6.89*	.032
Most efficient	6	5	16	2	8.22*	.016
Fewest errors	9	6	10	2	1.04	.595
Most frustrating	10	15	1	2	11.62*	.003
Initially easiest to use	5	4	17	2	12.08*	.002

* Significant at the .05 level.

6.89, p = .032). Sixteen subjects also perceived adaptable to be the most efficient condition ($\chi^2(2,27) = 8.22$, p = .016). Only one person found adaptable to be the most frustrating ($\chi^2(2,27) = 11.62$, p = .003). Seventeen people found the adaptable condition to initially be the easiest to use ($\chi^2(2,27) = 12.08$, p = .002). This was also reflected in additional comments, where several subjects noted that the adaptive condition was initially more difficult to use until they had become familiar with it. No significant deviation from the expected equal frequencies was found for Perceived Error Rate.

To test hypotheses H3 and H4 we compared the preference ratings using pre-planned comparisons. There was a significant difference between adaptable and static for the Overall Preferred dependent variable ($\chi^2(2,27) = 6.37$, p=.012), where 15 subjects specified adaptable as their most preferred menu type and 4 specified static as their most preferred. No other significant differences were found.

Additional comments that subjects included in the post questionnaire reflected a division between those who liked the adaptive menus and those who did not. Several people made positive comments; for example, saying the adaptive menus are fast "…before you know where things are really", referring to the fact that one has to familiarize oneself with the menu structure before the adaptable and static menus can be used efficiently. On the other hand, six people commented on the inconsistency of the adaptive menus, saying that this made the menus frustrating to use.

Summary of Results

We summarize our results according to our hypotheses:

H1: The adaptive menu was slower than the static menu. The adaptive menu was also slower than the adaptable menu, except when subjects used the adaptable menu first.
H2: The adaptable menu was not slower than the static menu, except when subjects used the adaptable menu first.
H3: The adaptable menu was preferred to the static menu, but not to the adaptive menu.
H4: Static was not preferred to adaptive.

DISCUSSION

Our work raises several interesting issues with respect to adaptive, adaptable, and static interaction techniques.

Need to Guide by Way of Example: Four of the five subjects who did not customize were given the adaptable condition first. This suggests that some users do not recognize the value of customizing, even when the customization mechanism is easy to use. Subjects who had first seen the static or adaptive menus recognized that placing the most frequent items near the top resulted in better performance. Previous work had suggested that users do not customize because the mechanisms are difficult to use [9]. Our work suggests that easy to use mechanisms are not sufficient. For effective customization, we may also need to guide users by providing examples.

Users Can Customize Effectively: Static split menus are the most efficient static menus documented in the literature. Optimal efficiency can be achieved for a static split menu when the actual frequencies of item selection are known in advance of the task, as was done in our experiment, but would be difficult to achieve in practice. The Block 1 data (Table 2) allows us to compare the static split menu to the non-customized version of the adaptable menu and shows that the static split menu was indeed 20% faster. (Recall that the non-customized version represents the static MS Word menus.) Therefore, the ability of users to customize their own menu (in two of the three orders) to achieve a result that was not found to be significantly different from the optimal efficiency of the static split menu is a strong result. Combining this with the finding that the adaptable condition was faster than the adaptive one (again for the same two orders), shows that adaptable interaction techniques can be effective and that more emphasis should be placed on them in interface design. In a previous field study, McGrenere et al. showed in that users were willing and able to customize their menus based on their function usage [10]; thus, the comparable findings we report here are not simply an artifact of our laboratory study design.

Majority of Users Want a Personalized Interface: Users value an interface that can be modified to suit their individual needs. Although the adaptable menu was preferred by the majority of subjects (55%), the adaptive menu did have support (30%). By contrast, only 15% of subjects wanted the static menu, even though it was the optimal split menu. This distribution differs somewhat from that of McGrenere et al.'s, who found that 65% of subjects preferred adaptable, 15% preferred adaptive, and 20% requested static menus (although no static variant was actually evaluated in their field study) [10]. The slightly greater support for an adaptive menu found here (30% vs. 15%) suggests that an adaptive split menu may be preferable to Microsoft's adaptive menu.

Perceived versus Measured Performance: The majority of subjects perceived themselves to be the most efficient with the adaptable menus, even though they were actually less efficient than with the static menus in one order, and were not significantly more or less efficient in the other two orders. This is a surprising result for which we have no concrete explanation. The results suggest, however, that pro-

viding users with control over their menus can lead to both better perceived performance and higher overall satisfaction.

CONCLUSIONS AND FUTURE WORK

Adaptive, or system-controlled interfaces, and adaptable, or user-controlled interfaces both provide dynamic approaches to tailoring interfaces to an individual user. We have presented a controlled lab experiment comparing the efficiency of these approaches using three types of split menu: static, adaptive, and adaptable. Results show that optimal static split menus are significantly faster than adaptive menus. When subjects were guided by example, i.e., they had seen the static or adaptive menus first, they were better able to understand the value of customization. Under those circumstances, the adaptable menus were faster than the adaptive ones; and the static and adaptable menus were not significantly different, showing that users can customize effectively. In qualitative feedback, we found that the majority of users (55%) preferred the adaptable interface to the other two, and ranked it first for perceived efficiency.

Our study found overwhelming support for personalized menu design. Even though more users preferred the adaptable menu to the adaptive menu, the users who preferred the adaptive one expressed strong support for it. This suggests that combining the two in a mixed-initiative design may be the best way to satisfy a wide range of users. Mixed-initiative interfaces, where the system and the user both control some of the interaction, have received relatively little attention in the research literature. One possibility would be to have the system periodically suggest additions/deletions to the top partition of a user's split menu.

Adaptable and adaptive approaches also need to be explored in the context of accessible interfaces for diverse user populations. Our sample was quite homogeneous; the generalizability to other populations is worth further study. For example, children may acclimatize easily to an adaptive interface, while users with cognitive disabilities may find adaptive interfaces entirely incomprehensible.

Finally, we expect to retest the hypotheses from this experiment in a field study to explore issues that would arise in a more naturalistic setting. Our work on menu design is promising; however, further study is required to see the extent to which adaptable and adaptive approaches generalize to other interaction techniques.

ACKNOWLEDGMENTS

We are grateful to NSERC for funding this research, to Kellogg Booth for early advice, and to Ron Rensink for generously providing access to the Visual Cognition Laboratory at UBC.

REFERENCES

1. Greenberg, S. *The computer user as toolsmith: The use, reuse, and organization of computer-based tools*. Cambridge: Cambridge University Press (1993).
2. Greenberg, S., and Witten, I. Adaptive personalized interfaces: A question of viability. *Behaviour and Information Technology*, 4(1) (1985), 31-45.
3. Horvitz, E. Principles of Mixed-Initiative User Interfaces. *Proc. CHI '99*, (1999), 159-166.
4. Hsi, I., and Potts, C. Studying the evolution and enhancement of software features. *Proc. Int'l Conference on Software Maintenance*, (2000), 143 -151.
5. Jameson, A., and Schwarzkopf, E. Pros and cons of controllability: An empirical study. *Proc. Adaptive Hypermedia 2002*, (2002), 193-202.
6. Kaufman, L., and Weed, B. Too much of a good thing? Identifying and resolving bloat in the user interface: A CHI 98 workshop. *SIGCHI Bulletin*, 30(4), (1998), 46-47.
7. Landauer, T. Chapter 9: Behavioral research methods in human-computer interaction. In M. G. Helander, T. K. Landauer and P. V. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (2nd ed., pp. 203-227). Amsterdam: Elsevier Science B.V (1997).
8. Linton, F., Joy, D., Schaefer, P., and Charron, A. OWL: A recommender system for organization-wide learning. *Educational Technology & Society*, 3(1), (2000).
9. Mackay, W. E. Triggers and barriers to customizing software. *Proc. CHI'91*, (1991), 153-160.
10. McGrenere, J., Baecker, R. M., and Booth, K. S. An evaluation of a multiple interface design solution for bloated software. *Proc. CHI 2002*, CHI Letters, 4(1), 163-170.
11. McGrenere, J., and Moore, G. Are we all in the same "bloat"? *Proc. GI 2000*, (2000), 187-196.
12. *Microsoft Office 2000 Products Enhancements Guide*. http://www.microsoft.com/Office/evaluation/ofcpeg.htm. Retrieved December 10, 2003.
13. Norman, D. *The invisible computer*. Cambridge, MA: MIT Press (1998).
14. Sears, A., and Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. *ACM TOCHI*, 1(1), 27 - 51.
15. Shneiderman, B., and Maes, P. Direct manipulation vs. interface agents: Excerpts from debates at IUI 97 and CHI 97. *Interactions*, 4(6), (1997), 42-61.
16. Thomas, C., and Krogsoeter, M. An adaptive environment for the user interface of excel. *Proc. IUI '93*, (1993), 123-130.
17. Warren, J. Cost/benefit based adaptive dialog: Case study using empirical medical practice norms and intelligent split menus. *Proc. IEEE AUIC 2001*, (2001), 100-107.
18. Weld, D. S., Anderson, C., Domingos, P., Etzioni, O., Krzysztof, G., Lau, T., et al. Automatically personalizing user interfaces. *Proc. 18th Int'l Joint Conference on Artificial Intelligence*, (2003).