What Role Can Adaptive Support Play in an Adaptable System?

Andrea Bunt, Cristina Conati, Joanna McGrenere
Department of Computer Science
University of Bristish Columbia
201-2366 Main Mall
Vancouver, B.C., Canada
{bunt,conati,joanna}@cs.ubc.ca

ABSTRACT

As computer applications become larger with every new version, there is a growing need to provide some way for users to manage the interface complexity. There are three different potential solutions to this problem: 1) an *adaptable* interface that allows users to customize the application to suit their needs; 2) an *adaptive* interface that performs the adaptation for the users; or 3) a combination of the *adaptive* and *adaptable* solutions, an approach that would be suitable in situations where users are not customizing effectively on their own. In this paper we examine what it means for users to engage in effective customization of a menu-based graphical user interface. We examine one aspect of effective customization, which is how characteristics of the users' tasks and customization behaviour affect their performance on those tasks. We do so by using a process model simulation based on cognitive modelling that generates quantitative predictions of user performance. Our results show that users can engage in customization behaviours that vary in efficiency. We use these results to suggest how adaptive support could be added to an adaptable interface to improve the effectiveness of the users' customization.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Interaction styles, Evaluation/methodology

General Terms

Human Factors

Keywords

adaptable interfaces, adaptive interfaces, customization, cognitive modelling, GOMS, user modelling, mixed-initiative

1. INTRODUCTION

As the functionality in software applications grows, graphical user interfaces become more and more complex. As a result, it is becoming increasingly necessary to help users manage this complexity. How to best solve this problem is a topic of debate. Some researchers suggest placing users in control of managing this complexity by making the interface *adaptable*, i.e., allowing users to customize the application to suit their needs. Others advocate making the interface *adaptive*, i.e., able to model the individual user's interests, preferences and usage characteristics to allow the interface to tailor itself to each user (see [17] for a discussion).

The debate between adaptable and adaptive approaches to interface design was quite prominent in the early 1990s, and seems to have settled into the Intelligent User Interfaces community favouring adaptivity, while others in the HCI community favour adaptable solutions. Neither of the two communities, however, has fully addressed the problems with each approach. The adaptive approach can suffer from some users feeling a lack of control over the process, a lack of transparency and a lack of predictability [3, 4]. As a matter of fact, there are very few formal user studies that indicate if and when adaptation is effective. As for the adaptable approach, which has also not been extensively evaluated, there are indications that users rarely customize (e.g., [10]). Furthermore when they do customize, it is not clear if they can do so effectively.

We take a different stance on the adaptable vs. adaptive debate. Rather than try to prove that one approach is better than the other, our goal is to integrate the two. We believe that a system should take into account characteristics of the users, the tasks they need to perform and the interface, and choose the right approach given the full context of use. If the user is able to customize effectively on his/her own, no system-initiated adaptation is required. If not, the system could intervene to provide assistance. Thus, we advocate a mixed-initiative approach to interface adaptation. While mixed-initiative paradigms have been investigated in the context of other aspects of human-computer interaction, so far there have been few attempts to apply it to the adaptation of interface elements.

Before building such a system, however, we need to gain a better understanding of the value of customization and of adaptive support for it. In particular, we need to understand 1) if customization is worth the effort, 2) if users can customize effectively, and if not, 3) what specific features the adaptive system should take into account to provide support for effective customization. This paper focuses on issues 1 and 3. In addition, we draw on previous results to discuss issue 2 – the adaptable interface designed and evaluated by McGrenere et al. [12, 13]. Their evaluation provided useful insights on how users chose to customize their interfaces when given an easy-to-use mechanism, but did not measure the efficiencies of these strategies. If we are to provide adaptive support to help users take full advantage of an adaptable interface, we need to be able to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IUI'04, January 13–16, 2004, Madeira, Funchal, Portugal.
Copyright 2004 ACM 1-58113-815-6/04/0001 ...$5.00.

identify when and why users are not able to customize effectively on their own. By comparing the efficiency of different customization strategies and relating these differences to factors such as user expertise and task composition, we are one step closer to creating adaptive support for customization.

One way to determine how user customization affects performance would be through a user study. Such a study, however, would have to involve both a large number of subjects (to get a range of customization behaviour) and multiple sessions (to give participants an incentive to customize and to see how customization helps with the given tasks as they are performed repeatedly). We have chosen to first gain insight into effective customization in a low cost manner by using a process model to simulate and compare different customization scenarios and strategies. The goal is then to use the insight gained from this exploratory analysis to inform the design of future user studies that can test more specific hypotheses. The process model we have developed relies on a well-established technique for interface evaluation known as Goal, Operators, Methods and Selection Rules (GOMS)[1]. We use GOMS modelling to look at two main aspects of customization strategy, when users should customize and how they should customize.

In the rest of this paper, we describe how our GOMS models were used to generate performance estimates of customization behaviour. We begin with a brief description of related work. We then describe the GOMS simulation models. Next, we illustrate the results of the simulation and we conclude by discussing the implications for intelligent adaptive interfaces.

2. RELATED WORK

Other work that has studied the trade-offs between adaptable and adaptive systems includes [12, 13] and [5]. Both studies, however, had confounding variables that preclude getting a definitive sense of when and why adaptivity is better than adaptability. The field study described in [12, 13] compared a prototype adaptable interface to the SMART MENUS adaptive interface of Microsoft Word 2000 (MSWord). The study showed that the majority of the participants preferred the control offered by the adaptable interface and made extensive use of the customization facilities. These results, however, cannot be used to conclude that adaptable interfaces are always superior. While the adaptable interface was carefully designed to be easy to use, the adaptive interface is known to have design flaws, mostly due to the unpredictability of its behaviour [4]. Furthermore, the conditions were not counterbalanced. The study in [5], which focused on the adaptation and adaptability of content rather than interface elements, presented anecdotal evidence that some subjects strongly preferred the adaptable version while others strongly preferred the adaptive version. But in this study, the adaptable mechanism was somewhat cumbersome, once again preventing a fair comparison between the two approaches. The fact that these two studies obtained somewhat differing results, could suggest combining the two approaches by carefully designing both the adaptive and adaptable mechanisms, and thoroughly evaluating the circumstances under which each mechanism is most effective.

There have already been proposals of systems that mix adaptation and adaptability (e.g, [2] and [16]). These systems often have some aspects that are adaptive and others that are adaptable, or the user can turn off or override the adaptive behaviour. There are two main differences between these systems and what we are proposing. First, the adaptation is related to the presentation of content rather than interface elements. Second, they do not guide or support user-initiated adaptation in any way.

One exception is FlexExcel [15], which provides adaptive support for customization in Excel. This support comes in two main forms: 1) when the user repeatedly invokes the same Excel function with the same parameters, the system suggests defining a new macro or short-cut; and 2) the system reminds the user of already existing adaptations. The system-initiated adaptation relies mainly on frequency counts. In an evaluation of the system, the authors found that users often had difficulty using the information provided by the system-suggested adaptations. We believe that this mixed-initiative approach is the right direction, and that the reason for these results is that more needs to be understood in terms of how the adaptations (user-initiated or system-suggested) affect performance and which user characteristics (other than frequency of use) should be taken into consideration.

3. GOMS ANALYSIS

To compare how different types of customization affect performance in terms of time on task, we use the Goal Methods Operators and Selection rules (GOMS) cognitive modelling technique [1]. GOMS models are a family of engineering models that are used to evaluate an interface design by predicting performance of simulated sequences of actions. Creating a GOMS model requires identifying the user's goals (i.e., tasks) in a given interface and the methods available to achieve them. Methods are decomposed hierarchically until they consist of primitive interface operators (e.g., keystrokes, moving the mouse from one target to another and clicking mouse buttons). These primitive operators have associated times, usually gathered through empirical studies.

Running a GOMS model allows one to simulate the set of interface actions that a user would take to perform a particular task. For example, to select a menu item, the user would perform the following sequence of actions: 1) find the menu heading; 2) point the mouse at the menu heading; 3) click the mouse button (at which point the menu items are displayed); 4) find the desired menu item; 5) point the mouse at the menu item; 6) click the mouse button to select the item.

GOMS has been shown to be particularly effective at comparing two or more interface designs [6]. This is exactly the goal we are trying to accomplish in this paper, where the different interface configurations we compare correspond to different customization behaviours.

While GOMS performance estimates are usually calculated by hand, we run our models automatically by using the GLEAN tool [7]. GLEAN takes as input a GOMS model written in a procedural-like language and a simulated interface written in C++. Using these two inputs, GLEAN executes the model and generates a predicted execution time, broken down by individual methods, based on established values for the primitive operators.

Next we describe the components of our GLEAN simulation. We also describe extensions we made to GLEAN in order to perform this work.

3.1 The Simulation

The GLEAN tool generates performance predictions by executing a GOMS model of user actions on a simulated interface. For the purpose of our experiments, the simulated interface is the adaptable version of MSWord proposed in [12, 13]. The interface includes a customization mechanism that allows users to maintain two versions of the MSWord interface: the *Full Interface* with all the available features, plus a *Personal Interface* that contains only a subset of features selected by the user. A toggle button allows users to switch back and forth easily between the two interfaces at any time.

The purpose of having two interfaces is to allow a user to work in a functionality reduced interface, but with access to the full func-

Task	Description	Summary of Relevant Actions	# of Task Features	Distribution	Total # of Features Invocations (Size)
Letter	A very basic editing task	Formatting the layout and font of parts of the text	8 items	across 3 menus	14
Report	A conference-style multi-sectioned document	Formatting section headings; parts of the text; setting the document layout; inserting page numbers; formatting references; language checking	16 items	across 5 menus	66
Table	A split cell table	Inserting a table; inserting additional rows and columns; and formatting the table	8 items	across 2 menus + 1 icon	10
Revisions	MSWord's revision tracking functionality	Inserting comments; accepting or rejecting comments	3 items	across 2 menus	9
Figure	Adding a figure	Inserting a figure from a file; adding a caption	3 items	across 2 menus	3

Table 1: The tasks simulated by our GOMS models.

tionality one click away. The customization mechanism was specifically designed to be as lightweight as possible. As a rough indication of the amount of effort required to customize, adding or removing a feature to/from the Personal Interface requires six button clicks. If the user adds or removes multiple features at the same time, each additional feature requires only two more button clicks (i.e., the customization overhead is four clicks and each feature requires two clicks). We chose to focus on this two-interface model because it has been thoroughly evaluated for usability. There are, of course, other forms of customization that we could have modelled, but they would first have to be thoroughly tested to avoid having lack of usability as a confounding variable in any study comparing adaptability and adaptivity.

Different tasks and different combinations of tasks are likely to benefit from different types of customizations. Our goal is to determine the efficiency of different customization strategies and how they depend of factors including user expertise, the user's combination of tasks, and individual task complexity. Thus, we built five GOMS models, all of which are based on tasks that users would realistically perform using MSWord. The five tasks are described briefly in Table 1. The tasks have been designed to vary in complexity, where *task complexity* is a function of the number of task features involved (the fourth column in Table 1) and the total number of feature invocations (*task size* – the last column in Table 1). The GOMS models of the five tasks contain only the menu and icon actions the user would have to perform to complete the task, and do not simulate any of the typing or cognitive work. This is based on the assumption that those components would be the same regardless of customization strategy. We are interested only in comparing differences owing to customization.

3.2 Factors Affecting Performance

At the level of GOMS primitive operators, the number of features in one's Personal Interface is likely to affect performance in at least two ways. One factor is mouse pointing time. It takes longer for a user to point to a menu item that is further down the list. GLEAN's primitive operator for mouse pointing addresses this issue by using Fitts' Law to calculate the time to point to a target [11].

The second factor likely to impact overall performance is visual search time. A menu with more items will take most users more time to search through. In addition, it will take most users longer to find the correct menu when there are more menu headings. The search time is also likely to depend on the level of user expertise, i.e. the user's familiarity with the interface [14]. The types of visual search documented in the literature include:

1. Exhaustive Linear Search [9]: The user examines every menu item in a linear fashion.
2. Self-Terminating Search [9]: The user examines every menu item but terminates the search when s/he finds the target.
3. Hick's Law [8]: The time to decide among the menu items is a logarithmic function of the number of alternatives.
4. Default: Any visual search operation is assigned a constant value [14]

GLEAN models only the Default search. It assigns a default value of 1.2 seconds to any visual search operation regardless of menu length. As we will discuss shortly, the different types of search strategies can be related to user expertise. Our goal is to study how user experience affects customization, and so we modified GLEAN so that it could simulate all four visual search strategies listed above. We then defined four expertise categories with respect to the visual search strategies. In the next section we describe how we defined these categories.

3.2.1 Relating Experience to the Visual Search Parameters

Norman [14] states that, with experience, users are no longer affected by the number of items in a menu; thus, the search behaviour of a highly experienced user can be reasonably approximated by the Default strategy. Landaurer [8] states that Hick's Law is applicable when users find the menu items easily distinguishable, when they know the menu item is present in the menu, and they have had substantial practice with it. Thus, this search strategy seems to indicate a user who is very familiar with the interface but not sufficiently experienced to be unaffected by the number of menu items. There is no work that explicitly relates usage of Exhaustive Linear Search and Self-Terminating Search to user expertise. It is reasonable to assume, however, that Self-Terminating would be a search strategy easier to employ by users who are more familiar with a target menu, since they would be able to tell when they have reached the item they are looking for without having to evaluate all items available.

We use the four different search strategies to identify four levels of expertise, ranging from Default to Exhaustive Linear Search in order of decreasing expertise. Exhaustive Linear Search, Self-Terminating Linear Search and Hick's Law include a parameter that can be related to the amount of time it takes the user to process an individual item (we call this parameter Time per Item). Time Per Item is likely a function of both a user's reading speed and familiarity with the menu items. Thus, we define our categories of expertise as follows:

- **Extreme Expert**: Default Search.
- **Expert**: Hick's Law with a Time Per Item of 0.15 sec.
- **Intermediate**: Self-Terminating Search with Time Per Item of 0.5 sec.
- **Novice**: Exhaustive Linear Search with a Time Per Item of 1.0 sec.

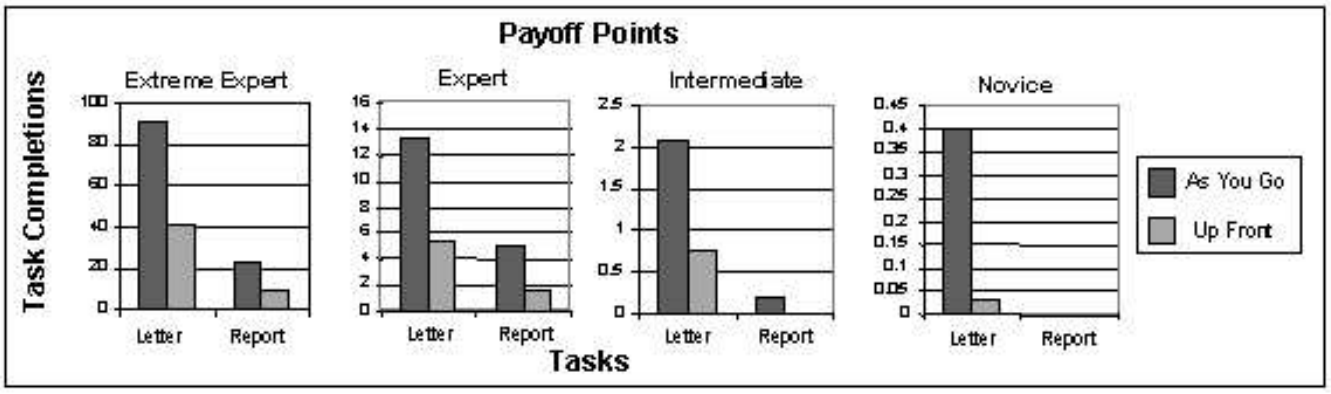

Figure 1: Experiment 1 results. The number of Task Completions it takes for the Personal Interface to outperform the Full Interface given the *Up Front* and *As You Go* strategies. For clarity of display, the graphs have different scales.

The values for Time per Item in each category are compatible with the values listed in the literature, but our experiments do not exactly replicate the conditions of previous experiments. As a result we choose values that are conservative estimates of the Time Per Item for each category. Our claim is not that these four categories perfectly model the behaviour of users with those levels of expertise, but rather that they provide four reasonable models that span user experience.

4. EXPLORATORY EXPERIMENTS

This section discusses two experiments we performed using the GLEAN simulation and the results that they generated. The first experiment addresses the issue of whether or not the overhead of customization pays off within a reasonable amount of time. It also compares customization strategies that vary in terms of *when* features are added to the Personal Interface. The second experiment is aimed at comparing strategies that differ in terms of *what* features users choose to add to their Personal Interface. We now discuss the details of these experiments.

4.1 Experiment 1: When to Customize

If users are able to customize their interfaces perfectly, do they perform their tasks more efficiently? A reduced interface is bound to be more efficient, but customization takes time. The goal of this experiment is three-fold. First, we want to see whether or not the overhead to perform customization pays off within a reasonable amount of time, assuming that the user has perfect foresight (i.e., adds exactly the features that s/he needs to the Personal Interface). Second, we want to compare the performance of two different customization strategies that differ in *when* customization is performed during the interaction. Third, we want to see how task complexity and user expertise influence the payoff.

To achieve these goals, we implemented GLEAN models for three customization strategies. These strategies are abstracted versions of those adopted by users in [12] where they were free to choose any strategy.

1. *Up Front*: The user adds all interface features that s/he plans to use before s/he starts the given task.

2. *As You Go*: The user adds the interface features incrementally (one function at a time) when they are first required to complete a given task.

3. *No Customization*: The user does not customize and uses the Full Interface.

The procedure for running the experiment is as follows. We ran the GLEAN simulation with models for the three customization strategies, two tasks (Letter and Report), and each of the four expertise categories. The assumption here is that the user repeats the same task over time and does not perform any other task. Letter and Report where chosen because they represent different levels of *task complexity* while being plausible tasks that one would repeat every day. To answer the question "can customization improve performance," we present the results in terms of the number of times the user would have to complete the target task (*task completions*) with the Personal Interface before it would outperform the Full interface, in terms of time to complete the task (*payoff*).

Can Customization Improve Performance? Our results indicate that customization can be worth the necessary effort. Figure 1 displays the number of task completions required for each expertise category before the given customization strategy pays off over *No Customization*. In the great majority of cases, users see a payoff within twenty task completions. In all cases, as expertise increases, the number of task completions increases. This indicates that users with more experience are less impacted by unused features in their interfaces and, therefore, must wait longer to see a payoff in customization. With larger task size (greater number of feature invocations), customization begins to payoff sooner because of the larger number of menu selections per task that are performed faster.[1] This results in a particularly dramatic effect for Intermediates and Novices, who often see the payoff immediately. The overall trends are not surprising, but the magnitude of the effect for Intermediates and Novices was unexpected.

If so, When? According to Table 2, based solely on the customization time, *Up Front* is always faster than *As You Go* – almost twice as fast in some cases. The one small benefit of the *As You Go* strategy is that users spend a longer percentage of their tasks with smaller Personal Interfaces. This saving, however, does not outweigh the extra customization time incurred by this strategy, since the *Up Front* strategy always leads to an earlier payoff than the *As You Go* strategy (see Fig. 1).

[1] In general we should also expect an effect based on the number of task features. In particular, the higher the number, the more delayed the payoff, because the size of the Personal Interface gets closer to the size of the Full Interface.

Task	Expertise	Up Front (secs)	As You Go (secs)
Letter	Extreme Expert	72	123
Letter	Expert	61	112
Letter	Intermediate	104	156
Letter	Novice	144	296
Report	Extreme Expert	137	250
Report	Expert	116	282
Report	Intermediate	205	318
Report	Novice	493	605

Table 2: The customization time for Letter and Report based on expertise and customization strategy.

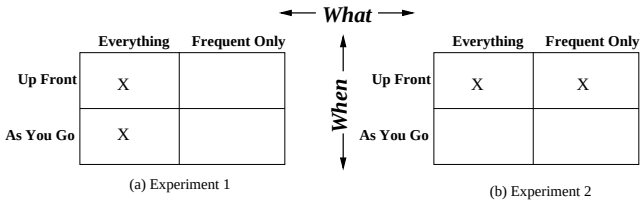

Figure 2: The two dimensions to customization strategy. The X's represent the combinations of strategies investigated in each experiment.

In summary, the results of this experiment show that customizing can be worthwhile, particularly for Novices. The most efficient time to perform the customization is *Up Front*, when the user can customize perfectly (i.e., has perfect foresight of the needed features). It may seem quite obvious that customization should save time by allowing one to work in a smaller interface, but we believe that a quantitative measure of this saving is important for determining whether or not it is worthwhile helping users to customize. We are aware that performance is not the only factor that triggers user customization, but we argue that knowledge of performance gains would make it much easier to motivate a user to customize a particular way.

4.2 Experiment 2: What to Customize

Unlike in Experiment 1, users are not likely to always be performing exactly the same tasks. Rather, they are likely to perform a combination of different tasks, with certain tasks being performed more frequently than others. This experiment is designed to explore which functions should be added to the Personal Interface under multiple task conditions, and how this depends on task combination and user expertise. Thus, while the first experiment addressed the *When* dimension of customization strategy, now we focus on the *What* dimension by analyzing the performance of two customization strategies that differ in the subset of functions that are added to the Personal Interface (see Fig. 2). As in the previous experiment, these strategies are based on customization behaviours observed in [12].

1. *Everything*: The user adds all features to the Personal Interface necessary for both the frequent and infrequent tasks.

2. *Frequent Only*: The user adds only those features necessary for the more frequently performed task, and then must switch to the Full Interface to perform the infrequent task.

To isolate the effect of the *Everything* vs. *Frequent Only* strategies, we assume *Up Front* customization. We restrict our investigation to combinations of two tasks, with the non-frequent task being performed once in the task sequence. The pairs of tasks (see Table 3) were selected based on *task composition*, which we define as a task's complexity (task size and number of task features) and the distribution of task features across the menus. We consider the distribution of features across menus (depth vs. breadth of the menus) because it affects interface complexity. Longer menus take longer to search through, but additional menu headings factor in all menu item searches as the user must first locate the appropriate menu heading and then the appropriate menu item.

Which customization strategy will be more effective involves tradeoffs between three main factors:

1. *Time to complete each instance of the frequent task.* With *Everything*, the more frequent task is executed in a larger Personal Interface than with *Frequent Only*, because the Personal Interface contains the features for both tasks as opposed to just the features for the frequent task. As a result, we expect the frequent task to be slower in the *Everything* condition. How much slower it is should depend on the complexity of the Personal Interface, user expertise and the composition of the infrequent task.

2. *Time to complete each instance of the infrequent task.* With *Everything*, the user can perform the infrequent task in the Personal Interface rather than in the Full Interface. Thus, the execution time of the infrequent task should be faster in the *Everything* condition. How much faster it is should depend on the complexity of the Personal Interface, user expertise and size of the infrequent task.

3. *Customization time.* The *Everything* condition requires additional features for the infrequent task to be added to the Personal Interface. The difference in customization time for the two strategies will depend on the number of features in the infrequent task and user experience.

Table 3 summarizes the variables that we manipulated in this experiment. The dependent variable is the total time necessary to complete the given ratio of tasks. As in the previous study, we used GLEAN to simulate the performance of the two customization strategies, with various combinations of tasks, task ratio, and user expertise. Tables 4 and 5 provide a high-level summary of our results. Table 4 indicates which strategy is more efficient for each combination without including customization time, while Table 5 includes the additional customization time required for the *Everything* strategy. For each task combination in Tables 4 and 5, where the most efficient strategy depends on user expertise or ratio (the last column in each table), Tables 6 and 7 indicate the task ratio at which the *Frequent Only* strategy begins to outperform the *Everything* strategy for each of the expertise categories. A range of ratios (e.g., (10-20):1) indicates that the exact ratio at which the *Frequent Only* begins to outperform *Everything* lies somewhere within that range. We now discuss the results in these tables by first ignoring customization time to isolate the effects of different interface configurations on performance. Later we examine how the additional customization time for the *Everything* strategy affects the results.

4.2.1 Comparing Strategies Without Customization Time

The results given in Tables 4 and 6 provide preliminary evidence of two factors that influence the efficiency of customization strategies when customization time is ignored.

1. The distribution of the features in the infrequent task, which determines the Personal Interface menus to which these items are added.

Independent Variable	Description	Levels
Strategy	Which functions are added to the Personal Interface	*Everything, Frequent Only*
Combination of Tasks	Frequent/Infrequent	Letter/Table, Report/Table, Report/Figure, Letter/Table, Report/Revisions, Letter/Revisions
Ratio	The number of times the user will perform the same frequent task for every one infrequent task.	1:1, 2:1, 3:1, 4:1, 10:1, 20:1, 100:1
Expertise	Categories of user expertise.	Extreme Expert, Expert, Intermediate, Novice

Table 3: A description of the independent variables in Experiment 2.

Task Combination (Frequent/Infrequent)	Everything	Frequent Only	Depends on Expertise or Ratio
Report/Figure	√		
Letter/Table			√
Report/Table			√
Letter/Figure			√
Letter/Revisions			√
Report/Revisions			√

Table 4: The most efficient customization strategy for each task combination without including customization time.

Task Combination (Frequent/Infrequent)	Everything	Frequent Only	Depends on Expertise or Ratio
Report/Figure	√		
Letter/Table		√	
Report/Table		√	
Letter/Figure		√	
Letter/Revisions			√
Report/Revisions			√

Table 5: The most efficient customization strategy for each task combination including customization time.

2. The complexity of the frequent task relative to the infrequent task.

The distribution of features in the infrequent task is an important factor since certain features, when they are added to the Personal Interface with the *Everything* strategy, have a greater impact on the execution of the frequent task than others. An infrequent task feature can have a large impact because 1) it gets added to a menu in the Personal Interface that is often used by the frequent task, or 2) it requires an entirely new menu heading. Adding a new menu heading is especially costly because it affects every visual search operation in the Personal Interface. When infrequent task features are intrusive to the frequent task, *Frequent Only* becomes the more efficient strategy at certain ratios and levels of expertise. To illustrate this effect, we can examine the Report/Figure and the Report/Revisions combinations in Table 4. In both combinations, the infrequent task contains only three functions, yet *Everything* is always the most efficient strategy for Report/Figure, while *Frequent Only* is more efficient at certain levels of expertise and ratios for Report/Revisions (see Table 6). The difference can mostly likely be attributed to the fact that the features in the Revision task add an entirely new menu heading, while two of the three Figure task features are added to menus that are rarely used by the Report task.

Let's now consider how task complexity, in conjunction with feature distribution, influences strategy efficiency. First, when an infrequent task feature is intrusive, the impact is greater for a larger frequent task than a smaller frequent task. For example, consider the Report/Table vs Letter/Table combinations in Table 6. The Table task adds another menu heading. The impact of this addition is greater for the larger task (Report) than the smaller task (Letter), since the larger task has more feature invocations to be impacted by the extra menu heading. As a result, *Frequent Only* becomes more efficient than *Everything* at lower ratios for Report/Table than for Report/Letter. In the absence of especially intrusive features, the number of features in the frequent task relative to the infrequent task is also likely to be a factor. If the Personal Interface contains a large number of features (for the frequent task), adding a small number of infrequent task features will not have much effect. This is not the case, however, if the Personal Interface is small to begin with. Both the Report/Figure vs. Letter/Figure combinations and the Report/Revisions vs. Letter/Revisions combinations illustrate this trend (see Table 6).

4.2.2 User Expertise

As in Experiment 1, the less experience a user has, the more s/he is affected by extra features. Thus, as user expertise decreases, Table 6 shows that *Frequent Only* usually starts to become more efficient than *Everything* at a smaller ratio. This is because of the higher impact of additional features on the speed of the frequent task. For example, we see that for the Report/Table task combination (without including customization time) *Everything* is the most efficient strategy for a Novice until s/he performs the Report task three times for every one Table task (3:1), as opposed to an Extreme Expert, for which *Everything* is always more efficient.

4.2.3 Including Customization Time

The discussion in the previous two subsections concentrated only on how the presence or absence of features in the two customization strategies affects performance. This ignores, however, the fact that *Everything* requires an extra amount of customization time that will vary according to the number of infrequent task features and the user's expertise. Tables 5 and 7 include this additional customization time. For many of the combinations, the extra customization does determine which customization strategy is more effective. *Everything* remains the most effective for the Report/Figure combination for all ratios and expertise categories because the extra cost is minimal given the few features in the Figure task. On the other hand, for three of the combinations that depend on expertise and/or ratio when ignoring customization time (Letter/Table, Report/Table and Letter/Figure), *Frequent Only* is always the most efficient strategy when customization time is included. Therefore, the performance savings from having the additional features present in the reduced interface is often not as great as the time needed to customize them.

4.2.4 Magnitude of Differences

In addition to determining which strategy is more efficient, our results also indicate how large the differences between the strategies are. The magnitude of the difference varies according to expertise and task combination. As an example, Figure 3 shows that

Task Combination (Frequent/Infrequent)	Extreme Expert	Expert	Intermediate	Novice
Letter/Table	*(>100):1	(10-20):1	(10-20):1	(10-20):1
Report/Table	*(>100):1	4:1	3:1	3:1
Letter/Fig	(20-100):1	(10-20):1	(4-10):1	4:1
Letter/Revisions	*(>100):1	(10-20):1	(4-10):1	(4-10):1
Report/Revisions	*(>100):1	(10-20):1	(10-20):1	(10-20):1

Table 6: Task ratio at which *Frequent Only* becomes more efficient than *Everything* for the task combinations in Table 4 when the most efficient strategy depends on ratio and expertise, and customization time is not included. The asterisk indicates that, for the ratios tested, *Everything* is always more efficient than *Frequent Only*

Task Combination (Frequent/Infrequent)	Extreme Expert	Expert	Intermediate	Novice
Letter/Revisions	1:1	1:1	2:1	3:1
Report/Revisions	1:1	1:1	1:1	(4-10):1

Table 7: Task ratio at which *Frequent Only* becomes more efficient than *Everything* for the task combinations in Table 5 when the most efficient strategy depends on ratio and expertise, and customization time is included.

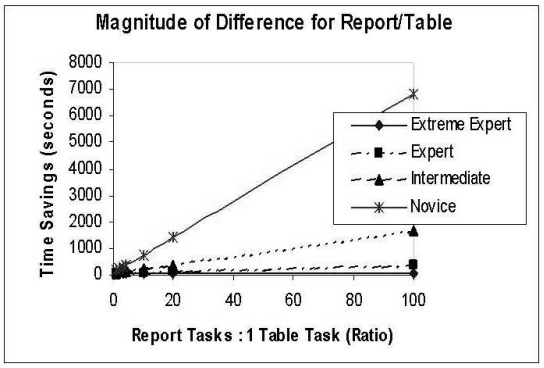

Figure 3: The magnitude of the difference between the *Everything* and *Frequent Only* customization quantities for the Report/Table combination for different ratios.

after having performed the Report task 20 times and a Table task once, a Novice would have saved 23 mins by using *Frequent Only*, while an Extreme Expert at the same ratio would save about 1 min (including customization time). At the same ratio (20:1) for the Letter/Figure tasks[2] a Novice would only save 7 mins while an Extreme Expert would still save about 1 min. This is because the few features in the Figure task do not add much to the complexity of the Personal Interface.

4.3 Cognitive Overhead

In the above experiments we did not consider the impact of two types of cognitive overhead involved with customization: 1) having to decide which features to include in the Personal Interface, and 2) having to figure out that a needed menu item is not in the Personal Interface and that switching to the Full Interface is required. The first would make the results in Experiment 1 somewhat less favourable toward customization. The second would make the *Frequent Only* strategy in Experiment 2 somewhat less favourable than our results suggest. We are fully aware of the importance of both of these cognitive overheads. Future work includes obtaining accurate estimates of their impacts so that they can be included in our model.

[2]Not shown in the figure due to lack of space

5. IMPLICATIONS

In this section we describe the implications of our research and the insights it has provided us into the adaptive vs. adaptable debate. Our main focus has been to investigate whether or not an interface should provide adaptive support for customization. This can be decomposed into three questions. 1) Is customization worth the effort in the first place? 2) If so, do users know how to customize efficiently? 3) If not, can adaptive support for customization be provided?

5.1 Is Customization Worth the Effort?

The results we have obtained indicate that if customization is done right, it has the potential to be extremely beneficial in terms of reduced time on task, even including the time it takes for users to customize. Most users will see performance benefits within the first 20 times they execute a task.

5.2 Do Users Know How to Customize Efficiently? Can We Provide Adaptive Support?

Combining our results with field study data from [12, 13], we see a number of ways in which users may not be able to customize efficiently. We also see the potential for adaptive support to help them overcome their difficulties. First, let's consider whether or not users know *when* to customize. Our results indicate that *Up Front* is much more efficient than *As You Go*, yet 70% of users in the field study did not engage in this type of strategy. Adaptive support could be used to encourage users to do as much customization as possible *Up Front*. It is possible, however, that in many situations, users will not be able to foresee all the features that will be needed. This is particularly true for Novices, who, as our results show, have the most to lose from inefficient customization. Therefore, adaptive support could be useful to efficiently implement an *As You Go* strategy, for example by recommending modifications at regular intervals by assessing user behaviour and estimating what features the user might need the most.

Second, let's consider whether or not users know how to customize (i.e., *what* features to add). The data from the field study indicates that 60% of users preferred to add all functions. Yet, our results show that when users perform one task infrequently compared to another, adding all functions is not always as efficient as adding only those from the frequent task. Whether or not the infrequently used features should be added depends on a number of factors, including the number of infrequently used features, where

these features are located in the menus, the ratio at which the infrequent features will be used compared to the frequent features, and the user's expertise. This is likely too many factors for a user to take into consideration when deciding what to customize, particularly for a Novice, who would again have the most to lose from inappropriate customization. Adaptive support could be used to help users be selective about which features they add to their Personal Interface.

Finally, in the field study, users never removed functions from their Personal Interfaces and those users who adopted the *As You Go* strategy often had trouble continuing to customize as the study progressed. Our results show that additional features not used on a regular basis can hurt performance. Their tasks may have evolved but their Personal Interfaces may not have. Helping users maintain their Personal Interfaces as their tasks evolve is yet another example of how adaptive support could help improve user performance.

5.3 Beyond Performance Data

In this paper, we looked at how customization strategies affect performance. Performance, however, is not the only factor that should guide adaptation. As indicated by McGrenere et al.[12, 13], there are a number of subjective factors that must also be taken into consideration to maintain a high level of user satisfaction. These factors include how users feel about full-featured interfaces versus reduced interfaces and how much control users desire over the content of their interfaces. Some users greatly preferred customizing their own interfaces, while others did not mind when the system adapted for them. An adaptive system should be able to balance these factors with performance considerations when providing the user with customization suggestions.

Complicating the matter is the fact that sometimes the magnitude of the difference between efficient and non-efficient strategies is quite small. Thus, the system will have to weigh the potential performance gain against the potential cost associated with a customization suggestion, both in terms of user satisfaction and the cognitive overhead involved in dealing with any advice from the system.

6. SUMMARY AND FUTURE WORK

Our results, combined with those from the field study in [12, 13], indicate that users may not always be able to customize efficiently. The details of the results suggest what specific problems user may have, and how adaptive support could help improve user performance in adaptable environments. The avenues that look the most promising are: 1) support for Novices, 2) helping users to customize as early as possible, 3) helping users to be selective about what they customize, and 4) helping users maintain their customized interfaces over time.

As future work, we plan to investigate ways to quantify the additional cognitive overhead that exists when users have to switch back and forth between the Personal Interface and Full Interface, the cognitive cost of deciding what features to include in the Personal Interface, and the cognitive overhead that would be introduced by any adaptive interventions. Second, we would like to explore to possibility of comparing our current results against results from an evaluation with human participants. Finally, we will use the results from our simulations to begin constructing the user model and adaptive algorithm that could provide tailored support for user customization. This will involve deciding how to best combine the subjective and objective factors that could influence effective customization, along with determining when and how the system should intervene.

7. ACKNOWLEDGEMENTS

We would like to thank Kasia Muldner, Steph Durocher, Rick Bunt and the anonymous reviewers for their helpful comments on drafts of the paper. We also thank David Kieras for his help in using the GLEAN tool and NSERC for supporting this work.

8. REFERENCES

[1] S. Card, A. Newell, and T. P. Moran. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., 1983.

[2] G. Fischer. Shared knowledge in cooperative problem-solving systems - integrating adaptive and adaptable components. In *Adaptive User Interfaces*, pages 49–68. Elsevier Science Publishers, 1993.

[3] K. Hook. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12:409–426, 2000.

[4] A. Jameson. Adaptive interfaces and agents. In *Human-Computer Interaction Handbook*, pages 305–330. Erlbaum, 2003.

[5] A. Jameson and E. Schwarzkopf. Pros and cons of controllability: An empirical study. In *Adaptive Hypermedia and Adaptive Web-Based Systems: Proceedings of AH 2002*, pages 193–202, 2002.

[6] B. E. John and D. Kieras. Using GOMS for user interface design and evaluation: which technique? *ACM Transactions on Computer-Human Interaction*, 3(4):287–319, 1996.

[7] D. E. Kieras, S. D. Wood, K. Abotel, and A. J. Hornof. GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of ACM UIST'95*, pages 91–100, 1995.

[8] T. Landauer and D. Nachbar. Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and the width. In *Proceedings of ACM CHI'85*, pages 73–78, 1985.

[9] E. Lee and J. MacGregor. Minimizing user search time in menu retrieval systems. *Human Factors*, 27:157–162, 1985.

[10] W. Mackay. Triggers and barriers to customizing software. In *Proceedings of ACM CHI'91*, pages 153–160, 1991.

[11] I. S. MacKenzie. Movement time prediction in human-computer interfaces. In *Human-Computer Interaction: Toward the Year 2000*, pages 483–492. Morgan Kaufmann Publishers Inc., 1995.

[12] J. McGrenere. *The Design and Evaluation of Multiple Interfaces: A Solution for Complex Software*. PhD thesis, University of Toronto, Toronto, Canada, 2002.

[13] J. McGrenere, R. Baecker, and K. Booth. An evaluation of a multiple interface design solution for bloated software. In *Proceedings ACM CHI 2002*, pages 163–170, 2002.

[14] K. Norman. *The Psychology of Menu Selection*. Ablex Publishing Corporation, 1991.

[15] R. Oppermann. Adaptively supported adaptability. *International Journal of Human-Computer Studies*, 40:455–472, 1994.

[16] K. Papanikolaou, M. Grigoriadou, H. Kornilakis, and G. Magoulas. Personalizing the interaction in a web-based educational hypermedia system: the case of inspire. *User Modeling and User-Adapted Interaction*, 13(3):213–267, 2003.

[17] B. Shneiderman and P. Maes. Direct manipulation vs. interface agents. *Interactions*, 4(6):42–61, 1997.