The Design and Evaluation of Multiple Interfaces: A Solution for Complex Software

by

Joanna McGrenere

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy

Department of Computer Science
The University of Toronto

Abstract

Computer software has become increasingly complex as advances in technology permit substantially more functionality to be provided to the user, a phenomenon which has led some people to describe today's heavily featured software as "bloated". Despite the prevalence of this trend, the impact of complexity on the user has received little attention in the research community. This dissertation describes research that addresses that problem.

Study One, a comprehensive study that looked at the experience of 53 users of Microsoft Word, showed that while many users would like to have unused functions "tucked away", most users were uncomfortable with the complete removal of unused functions. These findings suggested personalization as a promising direction for design and led to our Pilot Study which evaluated a multiple-interfaces prototype for Microsoft Word, where one of the interfaces was personalized to the user's individual needs. Results from that informal Wizard-of-Oz evaluation with 4 participants encouraged refinement of our prototype.

Study Two, a field study that included 20 participants, tested the effects of different interface designs on users' satisfaction and their perceived ability to navigate, control, and learn the software. There were two conditions: Microsoft Word with adaptive menus, and our user-adaptable multiple-interfaces prototype. Results showed that participants felt that they were better able to both navigate through the menus and toolbars and to learn with our multiple-interfaces prototype. There were also important differences in satisfaction and control with the new design.

This dissertation shows multiple interfaces to be a promising design solution to complex software. The novelty of our design is the *combination* of three design elements:

1) Multiple interfaces, one is personalized, one is the full set of functions, and switching between interfaces requires a single button click.
2) The personal interface is adaptable by the user with an easy-to-understand adaptation mechanism.
3) The personal interface begins small and, therefore, unless the user adds many functions, it will be a minimal interface relative to the full interface.

Important advances in the understanding of individual differences with respect to the perception of complex software and in the understanding of adaptive versus adaptable interfaces are made.

Acknowledgements

I am very grateful to my multidisciplinary supervisory committee.

Co-supervisors:

Dr. Ronald Baecker (Computer Science, University of Toronto) who agreed to supervise this research even though it was outside the scope of his main projects at the time. Among other things, Ron encouraged me to evaluate the prototype early in the design cycle, which became our Pilot Study, and provided insightful input into the Study Two protocol.

Dr. Kellogg Booth (Computer Science, University of British Columbia) who has proven that supervision at a distance can be done. Kelly was present throughout this research and at difficult times he always seemed to be there with sound judgment and direction. The spirit with which he supervised, his keen attention to detail, and his mega emails will not be forgotten.

Surrogate supervisor for Study One and committee member:

Dr. Gale Moore (Sociologist, Knowledge Media Design Institute, University of Toronto) who played an instrumental role in the early stages of this work and generously gave her time and expertise to mentoring a young academic. Together Gale and I shared both the joys and the challenges of doing cross-disciplinary research.

Committee members:

Dr. Mark Chignell (Mechanical Industrial Engineering, University of Toronto) who brought substantial expertise in the area of experimental design and data analysis. Mark made me welcome in his lab and allowed me to take part in his research group.

Dr. Ken Sevcik (Computer Science, University of Toronto) who always participated with a smile. Ken was thoroughly reliable and was never short of thoughtful comments.

Others who have directly contributed to this dissertation:

Dr. Mary Czerwinski (Cognitive Psychology, Microsoft Research) who facilitated access to a number of internal-use-only Microsoft tools and provided significant assistance with both the experimental design of Study Two and the resulting data analysis.

Dr. Saul Greenberg (Computer Science, University of Calgary) who acted as the external examiner at my Senate Oral and made valuable recommendations that improved the quality of this dissertation.

Dr. Ian Spence (Psychology, University of Toronto) who participated at my Senate Oral and gave his stamp of approval on the exploratory experimental approach taken in this work.

And lastly, those who contributed indirectly to this work:

Dr. Kori Inkpen who provided first rate accommodation at all the CHI conferences along with lots of late night conversation.

My fellow DGP lab mates who made a fun and collegial research environment.

Chris, Gwen, Tom, Scott, Tim, Melanie, and Joan who were always there providing much love and encouragement.

I gratefully acknowledge the sources of funding and in-kind support that made this research possible:

Communications and Information Technology Ontario (CITO) funded the Learning Complex Software project from 1998 to 2000. Our Study One was the first research to emerge from this cross-disciplinary project. Dr. Gale Moore of the Knowledge Media Design Institute is the principal investigator of this project.

IBM Toronto Lab provided 3 years of graduate funding in the form of an IBM Center for Advanced Studies (CAS) Fellowship.

Microsoft provided two software loggers (MSTracker and MS Instrumented Version) and an MSOffice expertise screening questionnaire. These tools were all developed for internal Microsoft use only but were made available for use in this research.

NSERC provided 2 years of graduate funding in the form of a Post Graduate Scholarship B (PGSB).

Substantial portions of Chapter Three and Chapter Five of this dissertation have already been published in the proceedings of the CIPS Graphics Interface 2000 conference and the ACM CHI 2002 conference. The questionnaire Experiencing Word Processing, copyright by Moore and McGrenere 1998, is included as Appendix F.

Table of Contents

CHAPTER 1 Introduction

The high-level goal of the research described in this dissertation has been to discover how people experience complex software and to use the findings to inform future interface design. We define complex software as software with many features. This includes lots of software application packages today.

1.1 Research Motivation

End-user applications have changed dramatically since the PC was introduced two decades ago. The sharp increase in raw compute power, and strong market forces, have resulted in desktop applications with sophisticated graphical user interfaces and with considerably more functionality than their predecessors. Productivity applications such as word processors compete in the marketplace in terms of the number of functions offered – a phenomenon that has become known as the *Feature War*[1]. The assumption seems to be that the greater the number of features, the more useful, or at least the more marketable the application. Paradoxically, as these applications have become more complex, enabling increasingly sophisticated operations, there has been rapid growth in the number of users, many of whom are unsophisticated. While some improvements have been made over the years in the field of Human-Computer Interaction, such as direct manipulation interfaces, the impact of this functionality explosion on the user has received little attention in the literature.

There has been some interest recently in the popular press and the computer literature in what has been termed “bloat” or “bloatware” [Munk, 1996; Kesterton, 1998; Kaufman & Weed, 1998a, 1998b; Cooper, 1999] and “creeping featurism” [Norman 1998]. The term “bloat” has

[1] We do not know who first coined the term Feature War. One example of its use is in a 1998 Macworld article by Snell [1998].

existed in the technical community for some time; software bloat has been defined as “the result of adding new features to a program or system to the point where the benefit of the new features is outweighed by the impact on the technical resources (e.g., RAM, disk space or performance) and complexity of use” [Online Computing Dictionary, 2001]. Creeping featurism is the tendency to complicate a system by adding features in an ad-hoc, non-systematic manner [Online Computing Dictionary]. One implication is that a bloated application is one in which there are a large number of unused features. In the popular press “bloat” has been used as a catch-all term suggesting software that is filled with unnecessary functionality. It always has a negative connotation but this characterization does not appear to be grounded in any systematic analysis of human experience [e.g., Gaudin & Nash, 1998; Do computers have to be hard to use? 1998].

What has been missing from the literature is both an understanding of how users actually experience complex software applications, and how understanding this experience can inform future interface design. Our initial motivation for this research was the assumption that the “average” user must be struggling considerably with complex software applications such as word processors and spreadsheets. But is this in fact the case and if so, is this primarily related to unused functionality? For example, do people feel overwhelmed by the number and variety of choices in the interface, and if so, how do they handle this? A related question is whether the interface can be designed to better accommodate the diversity of users?

1.2 Research Objectives and Overview

Our three main research objectives have been (1) to gain a systematic understanding of users’ experiences with complex software; (2) to move toward a new interface model that is derived from this understanding; and (3) to evaluate the new interface model in light of the problems that users experience.

The research conducted for this dissertation fulfilled all three objectives. Two formal user studies were conducted as well as one pilot study that preceded the second formal study. A prototype interface was designed, implemented, and evaluated.

Study One (Chapter 3) addressed our first main objective and provided a specific direction for our second objective. The goal of this study was to gain a systematic understanding of users' experiences with complex software from which we could elicit concrete ideas for design. The research questions included:

1) How do users experience functionality-filled software?

2) What factors impact a user's experience of complex software, for example, expertise, the number of functions known and used, and gender?

3) How do users learn complex software?

4) Does current interface design accommodate all users?

5) Are there identifiable groups of users that are satisfied and other groups that are not?

Study One involved 53 users of the application Microsoft Word, Office 97, and utilized social science methodologies to broadly capture the users' experiences of this complex software. A questionnaire was designed which included questions on participants' work practices, experience with writing and publishing, the use of computers generally, and the use of word processors specifically, as well as a series of questions for scale construction for the purpose of user profiling. A function identification instrument was designed to collect information on the use of and familiarity with Word's functions. Finally, an open-ended interview was conducted with each participant.

Among other things, Study One showed that individual differences with respect to the perception of heavily-featured software do exist. While many users would like to have unused functions "tucked away", most users were uncomfortable with the complete removal of unused functions.

As one might expect, the analysis of the data and interviews in the first study gave us a greater understanding of the ways in which people were actually experiencing a heavily featured application such as a word processor. This grounded our development of a design rationale, and many more questions had to be considered. For example:

6) Currently all users get the same interface (the “all-in-one” interface). Can the architecture for the user interface be better designed to accommodate a diversity of users?

7) How might today’s desktop applications be re-architected to support this type of diversity and what other benefits could be realized if it were?

8) Would all users benefit equally, or are there groups of users that would benefit more/less than other groups?

9) How do we evaluate and compare different interfaces for complex software?

These questions provided specific directions for our second and third main objectives, namely to move to a new interface model and to evaluate that model.

The concept of having multiple interfaces emerged from our Study One. The idea was to accommodate both the complexity of user experience and their potentially changing needs. Individual interfaces within this set would be designed to *mask complexity* and ideally to support learning.

For our Pilot Study (Chapter 4) we created our first multiple-interfaces prototype for Microsoft Word, Office 2000. To instantiate our conceptual design required detailed interface design and implementation. The prototype consisted of three interfaces between which the user could easily toggle: a minimal interface that contained a small number of the most frequently used functions, the default interface that contained all the functions that are found in an “out-of-the-box” version of Microsoft Word, and lastly a personal interface that was personalized to the user’s needs. The initial implementation was a Wizard of Oz prototype. The user could not personalize the interface him/herself; rather it was the researcher who made modifications to the personal interface based on the user’s requests[2].

[2] A recent workshop organized by Seffah, Radhakrishnan, and Canals [2001] and a publication by Menkhaus [2001] both use the term “multiple user interfaces”. Their use of the term is considerably different than ours. They use multiple user interfaces to refer to different interfaces or views for different devices used over the Internet for the same application or data repository, for example, an email application that has different interfaces for each of the desktop, mobile phone, and PDA client devices. By contrast, we use the term to

The Pilot Study was an informal evaluation of our initial implementation. Four users participated in this field study (one was the researcher). The prototype as well as a software logger was installed on each of the participants' machines for a period of 2 to 3 months. The researcher met with each participant periodically, made adjustments to the personal interfaces as necessary, and conducted informal interviews.

Key findings from the Pilot Study include that 3 out of the 4 participants preferred the prototype interface to the regular MSWord 2000 interface and that the individual differences identified in Study One appeared to play a role in these preferences. The minimal interface proved to be of little value; only one of the users made use of this interface. Through the Pilot Study we were also able to identify important logistical issues for running a field study with prototype software.

The results of our Pilot Study were promising and encouraged us to iterate on the design of our multiple-interfaces prototype and to perform a formal evaluation, namely Study Two (Chapter 5). The prototype was refined: the minimal interface was removed and an easy-to-use personalizing mechanism was added so that users could create their own personal interfaces. Twenty intermediate and advanced users of MSWord 2000 participated in this field study. There were two interface conditions in the study: Microsoft Word with adaptive menus, and our multiple-interfaces prototype. The prototype and a software logger were installed on each of the participants' machines for approximately 6 weeks and a series of online questionnaires was completed during this time. A final semi-structured interview was conducted with each participant to complete the study. The study tested the effects of the different interface designs on users' satisfaction and their perceived ability to navigate, control, and learn the software.

Study Two showed that participants felt that they were better able to navigate through the menus and toolbars and were better able to learn new functions with our multiple-interfaces

describe two or more interfaces that have different amounts of functionality for the same application on the same device. The term "multiple user interfaces" seems appropriate for either or both of these notions, since they address different dimensions of the problem of adapting the interface to the specific needs of the user and the context in which the user works. Nevertheless, throughout this dissertation we will use the term only to refer to our notion of two or more interfaces for the same device.

prototype. There were also significant differences in satisfaction and sense of control with our new design that relate to the previously mentioned individual differences.

In the Pilot Study and in Study Two, software logging technology was employed to capture the participants' interactions while they used the prototype software. Among other things we wanted to capture information about which functions were being used and how participants were personalizing their interfaces. Four loggers were investigated for this research and two were ultimately used. At the outset of our research we assumed logging to be a straightforward operation, however, through actual use of the loggers we came to realize that logging had considerable complexities that are as yet not fully explained in the literature. We document these complexities and our experiences with software logging in Chapter 6, and based on these we suggest desirable properties for a software logger.

One commonality that spans our studies was the use of Microsoft Word. Our intention has not been to generate an interface design solution specific to this one application. While it did make sense to focus on one application, the interface design that was prototyped and the results of the evaluations conducted are intended to generalize to other heavily featured productivity applications. Microsoft Word was deliberately chosen as our exemplar of complex software for two primary reasons: (1) word processing has been a canonical example in HCI research, so given our goal to investigate a feature-filled productivity application it was the obvious choice; and (2) the Microsoft word processing application was selected over other commercial word processors because of its dominance in the market place, which ensured relatively easy access to study participants, and because it was highly programmable through Visual Basic for Applications, which maximized the likelihood of our being able to create a prototype interface.

Figure 1-1 provides a schematic representation of the research discussed in this dissertation.

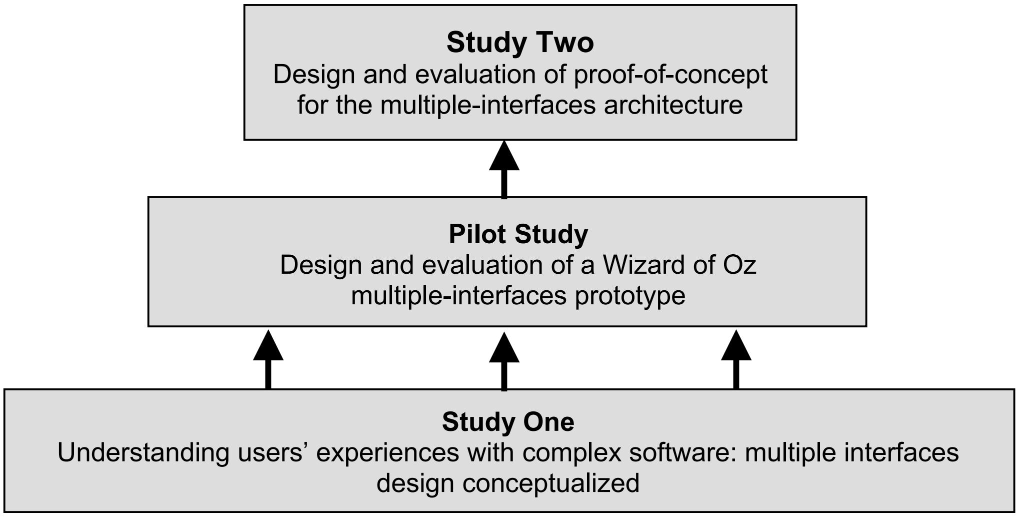

Study Two
Design and evaluation of proof-of-concept for the multiple-interfaces architecture
Pilot Study
Design and evaluation of a Wizard of Oz multiple-interfaces prototype
Study One
Understanding users' experiences with complex software: multiple interfaces design conceptualized

Figure 1-1: Dissertation research overview.

This dissertation is divided into seven chapters of which this introduction is the first. Chapter 2 reviews literature relevant to the use and design of complex software. Chapters 3, 4, and 5 cover Study One, the Pilot Study, and Study Two, respectively. Chapter 6 highlights our experiences with software logging technology and provides a set of general criteria against which loggers can be evaluated. Chapter 7 provides the conclusions of the research described in this dissertation and notes specific directions for future work.

CHAPTER 2 Related Work

In this chapter we highlight research that is related to the use and design of complex software. We start with research that documents how software has changed over time and then look broadly at how these changes have been experienced by the user, focusing in particular on how much functionality is actually being used and on the impact of learning this functionality. We then turn to design approaches for complex software. In particular, functionality blocking, menu design, customization, and intelligent interfaces, which include both intelligent agents and adaptive user interfaces, are all described.

2.1 How is Software Changing?

Anyone who has been exposed to the desktop computer over the last decade or two is aware that it and the software that makes it useful have been in a constant state of change. Although this is common knowledge, there has been little documentation of this reality in the literature.

Franzke and Rieman estimated that an office worker who relies primarily on three basic applications such as a word processor, a spreadsheet, and a graphics package, as well as an operating system, would be required to embrace a software upgrade on the average of every six months [1993].

With every upgrade inevitably comes new functionality. Hsi and Potts [2000] presented a view of feature evolution that was defined exclusively in terms of user-accessible features and concepts. They argued that there was no single model of this feature set and therefore generated a tripartite view:

1) The morphological view presents the structure that organizes an application's features. It consists of user interface elements including menu items, user input device

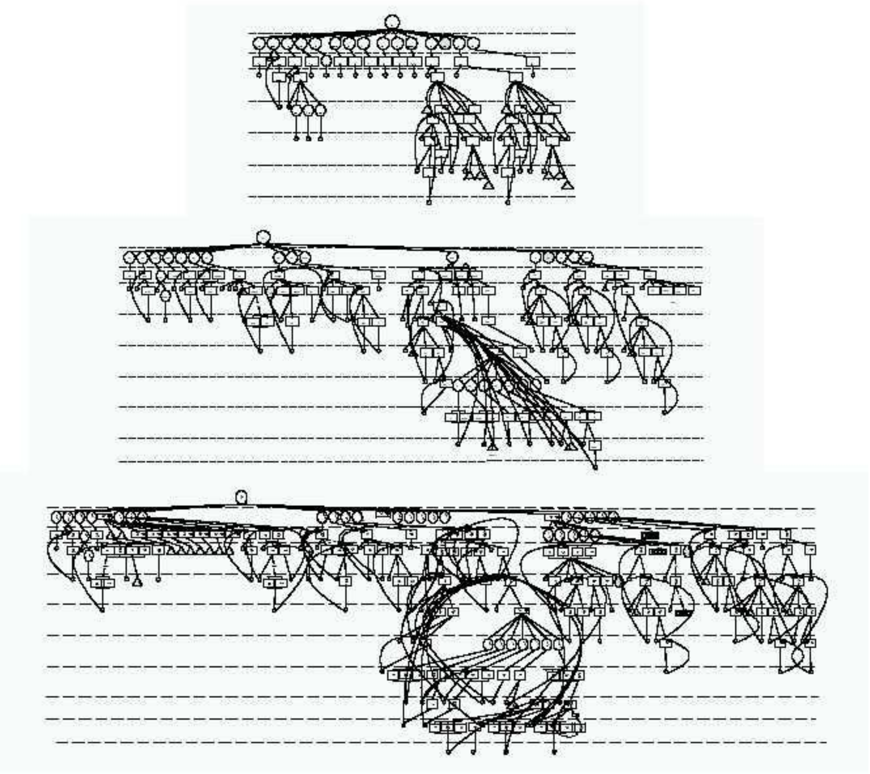


Figure 2-1: An overview of the graphs representing the Insert Menu morphology for MSWord 2.0, MSWord 95, and MSWord 97, respectively. (Copied from Hsi and Potts [2000].)

commands, and information displays. This view of the interface structure is independent of actual function.

2) The functional view enumerates all of the different operations that the features perform. These can be uncovered by traversing the morphological view.

3) The object view is the description of the subject matter or objects on which the features operate.

Hsi and Potts [2000] used these three views to study the evolution of Microsoft Word, in particular versions 2.0, 95, and 97. Two morphological trends were identified: (1) the morphological structure increased in size and complexity (see Figure 2-1), and (2) the types of primary interfaces changed, for example, MSWord 95 and 97 employed more mode shifts and toolbars to accomplish tasks. From a functional perspective, they found that MSWord underwent a steady, calculable growth in functionality – the total number of operations in each of the three versions was 311, 614, and 955 respectively[3]. The object view indicated an increase in the number of objects, however, the great majority of objects introduced in MSWord 97 were considered to have only peripheral relevance to what one might expect to find in a typical document. They noted that the evolutionary record of MSWord showed a large growth in morphology and a relatively smaller growth in the underlying objects and that "this could lead to user opinions that a product had become complex and 'bloated' far beyond its actual functional and conceptual growth" [p. 148].

2.2 Why Featurism? Why the Complexity?

The literature suggests a number of reasons why most software has become more-featured, namely, that features are needed in order to market the product successfully, that it is the programmers themselves (i.e., the technically-literate) who are deciding which functionality to include and are designing how that functionality is included, that the evolutionary process of software development does not easily accommodate global redesign, that additional features are needed in order for an application to integrate with other applications, and that usability guidelines favour giving users multiple ways to perform the same action. Each of these is discussed briefly below.

Marketing is driving featurism because features sell [Constantine, 1995]. It is somewhat a paradox; despite the fact that most buyers will never use many of the options, it is a comfort for these same buyers to know that the options are there just in case they may be needed someday. This can also be viewed as consumerism, i.e., that users want more for their money regardless of whether or not it will be used [Kaufman & Weed, 1998a, 1998b]. In the trade press software reviewers clearly focus on features by using tidy comparison tables that are packed full of different markers (checks, dashes, and circles with various fill patterns) which denote the extent to which a package has a certain capability. Vendors attempt to have the most checks (or filled circles) on the function list and consumers learn to discriminate between full-featured applications and those with fewer features.

[3] By comparison, Gibbs reported that MSWord 2.0 contained 311 commands, whereas MSWord 97 contained

The request for new features comes primarily from experienced users and these features are supplemented, designed, and implemented by programmers who are also experienced users [Computer Science and Telecommunications Board, 1997]. It is sometimes the case that programmers want to add easily coded features with little concern for the extent to which the features will actually be used. The mentality is that if a feature is easy to code, then the cost is low, and so even if it will only benefit a few users, it is worth adding it. In addition, programmers are creative people and may want to add a new feature because it is innovative or challenging to code [Kaufman & Weed, 1998a, 1998b].

It has been suggested that it is not the addition of features that has caused the complexity but rather the manner in which they have been added. The process of software engineering is evolutionary, so rather than maintaining a clear and consistent global design through versions of an application, it is more often the case that the software turns into a patchwork of parts and pieces and usability suffers greatly:

> Creeping featurism results from the slow accretion of capabilities and is reflected in a bumpy and irregular user interface marred by idiosyncrasies and special functions that seem to grow like warts or carbuncles in the oddest places. [p. 163, Constantine, 1995]

Additional features appear in software products for compatibility or integration purposes. This is exemplified in product suites such as Microsoft Office which give the user the ability to load a file from one Office product into another [Kaufman & Weed, 1998a, 1998b].

Lastly, new features may appear for what is considered to be a usability reason. It is sometimes thought that if users are given multiple ways to do the same thing, that the usability of the system is enhanced [Kaufman & Weed, 1998a, 1998b].

Thus the functionality explosion that has taken place is largely attributable to marketing forces and the resulting features are primarily targeted at and designed for the sophisticated user. The needs of other users don't appear to have been considered.

1,033. However, he provides no basis for his counts [1997].

2.3 What is the User's Experience of Complex Software?

Software has been under a constant state of evolution and the drive to include more and more features appears to be driven largely by forces other than the users' needs. Despite this, there has been minimal research to date that specifically addresses how users are coping with all this functionality; it doesn't appear as though anyone has taken on this issue directly. There have, however, been some accounts that are relatively informal and qualitative in nature and also some experimental research addressing the user's ability to learn functionality-filled software. Through the use of software logging a number of field studies have also uncovered how much of this functionality is actually being used. We briefly review each of these areas separately and then provide an overall summary at the end.

Constantine [1995] informally likened featurism to a disease:

> Word processors, and a growing legion of our most important software tools, have become victims of creeping featurism, a serious malady of user interfaces that strikes software in its prime and can, if left unchecked, cripple the user. Untreated, creeping featurism can leave users with an agoraphobic response to large, open dialogue boxes, or even with a lingering fear of unknown menus. [p. 162]

Others have alluded to these complex systems as being "nightmarish" for the user:

> Our present systems have come to be as large, complex, and nightmarish as the mainframes they first displaced (mainframes have become larger still; but most computer users don't have to deal directly with them on a daily basis). [Raskin, 1997, p. 99]

Survey research by Munk [1996] suggested that having powerful PCs filled with heavily featured software can reduce users' productivity: "Too much horsepower on the desktop can have the perverse effect of cutting productivity." For example, she reported that a survey of 6,000 workers conducted by SBT Accounting Systems in San Rafael, California, found that users spent five hours a week "futzing" with their PCs. It is not entirely clear what constitutes futzing, but she included the following as significant time wasters: waiting for programs to run, reports to print, repair men to show up, technical support folks to pick up the telephone, and organizing and clearing out cluttered disk storage.

Microsoft reported in a workshop an unpublished study the goal of which was to gain an understanding of users' perception of bloated software [Kaufman & Weed, 1998a, 1998b]. Twelve members of a focus group were asked to define "bloat". They were asked to complete the statement *"My software feels bloated when…"* The sample included intermediate and expert users. There was no information provided about how the group was chosen, but the results are intriguing. The study suggested that users can be categorized into one of two Profiles, A or B, depending on their perception of software as "bloated" or not. Profile A users prefer software that is complete, they will stay up-to-date with upgrades, they assume that all interface elements have some value, and they blame themselves when something goes wrong or when they can't figure out how to perform a specific task. Alternately, Profile B users prefer to pay for and use only what they need, they are suspicious of upgrades, they want only the interface elements that are used, and they blame the software and help system when they can't do a task. According to the study, a user's profile was independent of expertise. We will comment more fully on these findings later in this dissertation. In particular, we extend Microsoft's work by identifying three profiles: the feature-shy, the feature-neutral, and the feature-keen.

2.3.1 What is the Impact on Learning?

A number of studies have shown the impact of complexity on the user's ability to learn software. Before describing these studies we briefly discuss the issue of learning in the context of computer technology.

There is no single definition of learning, however, Nilsen et al. [1993] observed that the HCI literature often adopts skill acquisition as representative of learning. There have been a number of different approaches to learning complex systems documented which include exploratory learning, learning through transfer of prior knowledge, formal training, learning through user support provided with a system (documentation, online help, tutorials, animations, and demonstrations), and learning through assistance from colleagues and friends. Research shows, however, that learning through exploration is the preferred method

[Howes & Payne, 1990; Rieman, 1996; Carroll & Mack, 1984] although it may not necessarily be the most efficient or effective method of learning how to use a system. The interactivity of applications encourages exploratory learning but it also allows for unproductive exploration and thus poor learning [Trudel & Payne, 1995]. Exploratory learning can be enhanced when the learner is forced to reflect on his/her interactions and when the exploration space is constrained. Reflection can be forced by limiting the amount of interaction the user can have with a system or by making the interaction more costly [Svendsen, 1991]. For a broad treatment of learning in the context of computer technology, the reader is encouraged to see the review by McGrenere, Baecker, & Booth [1999].

Carroll and colleagues [Carroll & Carrithers, 1984a, 1984b; Carroll & Mack, 1984; Mack, Lewis, & Carroll, 1983] created what they called a *Training Wheels Interface* for a commercial word processor. Their interface essentially blocked some functionality and error states, so that when users tried to select a blocked function, a message was displayed that indicated that the function was unavailable in the Training Wheels Interface. Two studies were conducted each with 12 novice users that compared the training wheels system (TW) to the complete system (CS). The first study found that the TW users could complete a simple word processing task 21% faster and spent significantly less time ($p<0.005$) recovering from errors when using the altered system compared to the CS novices. *Error* was defined here as a departure from the create-and-print action path and *recovery* was defined as a subsequent return to that path. In a post-session comprehension test of word processor basics the TW users performed significantly better than the CS users ($p<0.05$). And in a questionnaire designed to reveal attitude toward work, the TW users scored significantly higher than the CS users ($p<0.05$). It was conjectured that because the TW users were more successful, they may have felt better about themselves and about work in general. The second study was almost identical to the first and most of the mean differences in the second study closely tracked those of the first study. Two differences in the second study were that the TW users committed significantly fewer errors ($p<0.005$) but there were no significant differences in the post-tests on comprehension and work attitude.

Franzke and Rieman [1993] conducted a study with 12 users who had an average of two years experience with Macintosh computers. The study compared how long it would take users to learn how to create a default graph in two different versions of a graphing package. They found that it was significantly faster ($p<0.05$) using the earlier version of the package than the later version. The graphing task specifically required the navigation and completion of a particular dialog box that roughly tripled in features between the two versions tested. Clearly, if the experimental task had required the use of the features available only through the newer version's dialog box, then it would probably have been the case that using the newer version would be faster. (There would, of course, be no comparison at all if it were impossible to accomplish the task in the earlier version.) This raises the question of how much of the functionality made available is actually used.

Franzke [1995] investigated the impact of the number of interface objects on a user's ability to find the appropriate object. She conducted an experiment with 76 reasonably experienced users (average of 2.8 years of Mac experience and average familiarity with three Mac applications) that included two trials. It was found that as more objects were displayed at a given time, it took significantly longer for the users to locate the object needed. But this main effect interacted with trial. Thus the action times during the first trial (the exploration trial) were more affected than those during the second trial. During the second trial, the number of objects did not matter as much. There was a three-way interaction between the number of objects, the quality of an object's label, and the trial such that the search time for poorly labeled objects was significantly worse when there were many objects to search in trial one. If the label quality was good there was no effect of the number of objects on display. There was another triple interaction observed between the type of interaction, the number of objects on display, and the trial. If there were many objects to search, action times during trial one were inflated especially for interactions that were difficult to discover. Such interactions refer to, for example, clicking on a dialog-box button, editing text, using a radio button, single clicking on a graph object, etc. In particular, the types of interactions that fall under the category of direct manipulation required more time.

In terms of total functionality usage, Nilsen et al. [1993] conducted a longitudinal study of people using a commercial spreadsheet package. The study lasted 16 months and included 26 subjects who were followed during their learning of a popular spreadsheet program, Lotus 1-2-3. The subjects were all new MBA students who were expected to use Lotus 1-2-3 for school-related work. Each subject was tested in a laboratory setting at the outset and then was subsequently given an almost identical test at each of three equally spaced intervals, thus there were four testing sessions. The tests consisted of tasks that required knowledge of basic Lotus 1-2-3 functionality. The results showed that 14 of the 26 subjects (54%) were able to complete all the tasks in the test, 7 (27%) were able to complete most tasks but made some errors, and the remaining 5 subjects (19%) committed multiple errors which displayed a basic lack of knowledge about Lotus 1-2-3. Nilsen et al. concluded that people only use a subset of an application's functionality and they don't often master even this subset, let alone learn and master all of the functionality.

Nilsen et al. [1993] reviewed the HCI literature and found that studies of the growth of software skill show "that people attempt to fully master the current task-related skill before moving on to more complex, advanced skills or those relevant to other tasks." This suggests that users prefer to feel like an expert on a functionality subset rather than like a novice on the total functionality. When applications provide users the choice to operate at different expertise levels within a complex system, Shneiderman [1997b] claims that users are content to remain experts at level one, rather than dealing with the uncertainties of higher levels. The only systems he cites as giving this choice are computer/video games and HyperCard. (Although most systems permit some form of customization, Shneiderman is specifically referring to systems that allow the user to set a level and then availability of functionality is strictly based on the level selected.)

Dillon reported on a longitudinal evaluation of two different interfaces for the same information retrieval system [2001]. One interface was designed according to current interface design guidelines (the friendly interface) and the second interface was designed to specifically break every one of the design guidelines (the unfriendly interface). One group of users was exposed to the friendly interface for 11 sessions (one session per day) and had to complete a series of tasks. A second group of users was similarly exposed to the unfriendly interface. The results showed that although there was a significant difference in performance in the earlier sessions (with those in the friendly interface condition performing significantly better than those in the unfriendly interface condition), by the latter sessions the two groups of users were performing equally well. From this Dillon concluded that people will adjust to technologies if they persist with them.

There were two additional twists on Dillon's experimental design [2001]. For the eleventh session a form of stress was introduced into the search task – each user was told that his/her individual performance was the worst of all the participants and that it was necessary for an observer to watch directly over the participants' shoulders and time the tasks in order to determine where the performance problems were occurring. Interestingly, the performance of the users of the unfriendly system decreased significantly whereas that of the users of the friendly system decreased only very slightly. The second twist was that for a twelfth (and final) session users swapped interfaces; those who had used the unfriendly interface switched to the friendly interface and vice versa. The users who had used the friendly interface for the first 11 sessions had no significant change in performance when they switched to the unfriendly interface. Those who originally used the unfriendly interface, however, had significantly poorer performance when they switched to the friendly interface. From this Dillon concluded that those who had been first exposed to the friendly interface were able to develop a good conceptual model of the system that enabled them to adopt a different, albeit worse, interface for the same system. Those users who began with the unfriendly system never developed a satisfactory conceptual model and therefore struggled to adopt a different interface even though it was a significantly better interface.

2.3.2 What Functions are Actually Being Used?

Another perspective of users' experience of functionality-filled software is gained by looking at the proportion of functions that are being used from the total available. Research studies investigating command usage in the context of users carrying out their normal everyday tasks consistently show that the majority of the time users of complex software use very few of the commands available; informally this is called the 80/20 rule. The majority of this work was done in the 1980s and looked at the Unix operating system and Unix tools (e.g., text editors). Two studies that are representative of this literature are described below. In addition, the results from a more recent study of the commercial word processor Microsoft Word are discussed.

Hanson, Kraut, and Farber [1984] collected usage data for 16 users over 2 weeks through software logging while the users carried out their normal daily tasks. Of the 400 Unix commands available they found that the 20 most frequent commands (5%) accounted for about 70% of the usage and that the rank frequency distribution of the data could be represented as a Zipf [4] curve. They indicated that there are many reasons why most of the commands were used so infrequently: redundancy among commands, difficulty of use, or the infrequent need for them. Hanson et al. noted that:

> …whatever the reasons, the uneven distribution of command use suggests that computer systems should find ways to increase the prominence and ease of access to frequently used commands. The traditional command organization, in which a large number of commands is as prominent as a smaller number of frequently used ones, is likely to intimidate new users, hinder the learning of the command set, make identification and location of important commands difficult, and lead to erroneous command entry. [p. 45]

Greenberg [1993] studied the command usage of 168 Unix users over a 4-month period looking specifically at command reuse. His work supported some of the earlier Hanson et al. findings [1984] and extended the analysis to look at individual user behaviour and the behaviour of groups of similar users. Greenberg found that the rank frequency distribution of command usage by groups of like and unlike users was approximated by a Zipf distribution. For the most part, the actual rank order of commands differed between groups. Surprisingly, the overlap between the command vocabularies of different users was minimal, even for users in the same workgroup who performed similar tasks and who had similar computer expertise. Individual users had small command vocabularies, and the acquisition of new commands was slow and irregular. Consequently, while the Zipf model may represent aggregate behaviour, it may not be an accurate estimate of an individual's behaviour.

[4] Zipf's law is the observation that frequency of occurrence of some event (P), as a function of the rank (i) when the rank is determined by the above frequency of occurrence, is a power-law function $P_i \sim 1/i^a$ with the exponent a close to unity. A comprehensive list of Zipf's law references can be found on the Rockefeller web site [References on Zipf's Law, 2001].

Linton, Joy, and Schaefer [1999] logged data from 16 users of Microsoft Word 6.0 over a period that ranged from 3 to 11 months per user. For that version of MSWord they counted 642 different commands and found that only 152 of these commands were invoked during the logging period. The total number of command invocations was 39,321. On average each user used 56 different commands (8.7%, SD=3.9%) throughout the logging period and applied 25 (3.9%) different commands in an average month. The top 2 commands accounted for 25% of all use, the top 10 commands accounted for 80% of use, and the top 20 accounted for 90% of use.

2.3.3 Summary

In terms of understanding the user's experience of complex software we have found informal accounts that referred to complex software as nightmarish and likened it to a disease. High-level survey research found that heavily featured software can actually reduce users' productivity. Others have found that there are individual differences with respect to how the user perceives complex software – some users embrace the latest upgrade and the new functionality that it brings whereas others shy away.

A number of experiments have shown that users perform better initially with less functionality and one study showed that after more than 1 year of exposure to Lotus 1-2-3, approximately 20% of the highly-educated participants could not even perform basic tasks. Users prefer to master the current task-related skill before moving to a more complex skill and the relevant functionality for that skill. Yet, when users are given a sufficient amount of time to use and learn an application, they have a large capacity to persist and succeed in the face of poor design.

Looking at command repertories we found that the majority of the time users do use very few of the available commands but there tends to be minimal overlap between the repertories of individual users even when they are performing the same tasks.

What is missing from the literature is a comprehensive approach to understanding the user's experience. Counting commands provides some insight but it does not tell the whole story. For example, command usage does not indicate the impact on the user of having so many commands to choose from and, by extension, whether or not users embrace new functionality. The Microsoft profiling approach indicates that there are individual differences at play but this is not set in the context of what commands are actually being used. Are the users who shy away from upgrades the ones who use relatively few functions? The studies on learning certainly provide a strong indication that users are better off from a performance perspective with only the functions that they need, but this does not take into account users' affective response to the technology. For example, do users feel more empowered by having so many functions to choose from?

2.4 Design Approaches for Complex Software

There has been some effort to design systems in order to alleviate complexity but the results are far from conclusive. Approaches include blocking off un-needed functionality, effective menu design that allows users to locate desired items, and supporting the user's ability to customize his/her own interface in a way that is suitable to his/her needs. Another general approach has been to design more intelligent user interfaces – intelligent in the sense that they rely on artificial intelligence to offload some of the complexity from the user.

2.4.1 Functionality Blocking

Functionality blocking, as the name suggests, attempts to reduce complexity by blocking off or hiding significant amounts of functionality from the user.

When users of varying levels of expertise must be accommodated in one system, Shneiderman [1997a] recommends a *level-structured* approach (sometimes called a *layered* or *spiral approach*[5]). He says that novices should be taught a minimal subset of objects and actions with which to get started because they are most likely to make correct choices when they have only a few options and are protected from making mistakes. This is in essence a

[5] This approach is based on the spiral model for software engineering [Boehm, 1988], although there is a similar spiral model of learning that progressively discloses more and more detailed aspects of a topic as students gain mastery of the material.

form of functionality blocking where the functionality is layered. Unfortunately, Shneiderman doesn't suggest specific design guidelines to accomplish this design strategy.

The Training Wheels interface by Carroll and Carrithers' [1984a, 1984b], first introduced in Section 2.3.1, is a classic example of a level-structured approach where users only have to deal with a certain amount of functionality at any given time. They showed that by blocking off all the functionality that was not needed for simple tasks, novice users were able to accomplish tasks both significantly faster and with significantly fewer errors than novice users using the full version. This design has the user progressing through two distinct phases. After the first phase the training wheels are removed, launching the user into the full system[6]. Research issues related to the transition from the first phase to the second phase have not been further investigated.

Another form of functionality blocking appears in the commercial domain. This is the simultaneous availability of versions that differ only by the amount of functionality offered, for example, the popular email program Eudora Pro and Eudora Light. This product versioning appears, however, to be related more to a business cost model rather than an attempt to model user needs. In the case of Eudora, the Light version is free of charge. The transition from one version to the other is done explicitly by installing the desired version.

The functionality blocking approach appears to be promising, however, the transition between blocked and unblocked states and the predetermined yet unchangeable nature of the blocked states are issues that have not been dealt with adequately, which is why it has remained mostly a theoretical/research design.

2.4.2 Menu Design

The design of menu systems is highly relevant to the design of complex software and to the specific research direction taken in this dissertation. Research in menu design has been

[6] Carroll and Carrithers' Training Wheels does for the user interface what scaffolding does for content. Scaffolding is a technique used in learner-centered design [e.g, Soloway, Guzdial, & Hay, 1994; and Quintana, Carra, Krajcik, & Soloway, 2001]. The clear parallel between use of educational software to learn content and the learning that takes place when a user interface is being used is further evidence for this similarity.

concerned with the question: how should functions be positioned within a menu system for most effective and efficient use? There was a significant amount of research conducted on menu design in the HCI community in the 1980s that has been synthesized in a comprehensive book on the psychology of menu selection by Kent Norman [1991]. Here we summarize some of his key findings and recommendations; our intention is not to provide a summary of menu design in its entirety but rather to mention those findings that are most relevant to our own research.

1) Menus may display only the commands that are available in the current context or they may display the full set of commands with the unavailable items grayed out. The advantage of retaining the grayed-out items in the list is that the presence and location of items are fixed. Users are exposed to all the items and are informed as to which are available. What they may not know, however, is how to get to the context that will activate desired items.

2) Closely related to the above, menu items may be switched or rephrased depending on the current context. In one context the menu may display the item "Open File" and in another it may change the item to "Close File." Although rephrasing menu items reduces the number of menu items, it may be problematic in a situation where the user is looking for a recalled menu item without being able to recall the required context. There is no research literature that evaluates the effectiveness of rephrasing menu items.

3) Users go through a scanning and recognition process when selecting an item in a menu. Response time for menu scanning is a function of the number of items scanned and the time required to scan each item. However, these processes tend to become automatic with extensive practice. When users routinely proceed through the same processes of scanning and selection they can develop to a point where their response times are no longer affected by the number of items or number of targets.

4) There are two straightforward tradeoffs when considering the amount of information to include in a menu system. The first is the amount of information vs. scan/reading time. The more information, the more difficult and time-consuming it is to search for

target information and to read the information. The second is the amount of information per menu vs. the number of menus. Given a fixed number of menu items, the shorter the menus, the greater the number of menus required.

5) Applying some organizing principle (e.g., alphabetic, chronological, frequency of use[7], categorical) to the ordering of items within a menu facilitates searching for an item. However, even if a menu is randomly ordered, frequent users of a menu system will learn the spatial location of items so long as the list remains constant.

6) In order for users to make use of spatial context, the position of menu items should be standardized. A study by Teitelbaum and Granda [1983, cited in Norman, 1991] found that the time it took to answer questions about information on the screen was longer when the location of information was inconsistent than when the location was constant. In addition, over time users' response times improved with practice when the information was in a constant location, which indicated that users learned where information was located. Response times did not improve when information was presented in random locations. Norman concludes that the importance of consistency in screen and menu design cannot be overemphasized and that it is imperative that users develop expectations about the location of information to avoid having to repeatedly search for information.

7) If the occurrence of items in a menu varies such that sometimes when the menu is invoked the items are available and other times they are not available, their position should at least remain constant. Consequently, the method of graying out unavailable items in a list would be superior to deleting them and shortening the list. The overriding guideline is to maintain absolute positional constancy, not just relative position.

8) If a designer knows that some items will be more frequently accessed than others, they should be placed in a broad menu at the top positions of the pull-downs to

[7] This is a static menu that is organized with the most frequently used items at the top. This is not to be confused with a dynamic menu that adapts based on an individual user's frequency of menu item use.

minimize access time. So frequency of use should be the organizing principle, at least for the first few items in each menu.

9) Novice users spend most of their time searching for the location of an item, so frequently used items should be placed in the pull-down locations searched first, which is often the top left-most pull-down menu. Experienced users spend most of their time in cursor movement and so it doesn't matter in which pull-down a frequently used item is located, but it is important that it is in a top position.

10) Norman cautions that although the concept of menus that dynamically adapt based on usage patterns is appealing because of their potential to enable faster access to menu items, they may carry a "fatal flaw":

> While the system "learns" and adapts the menu to usage patterns, the user is kept guessing as to the location of an item. High-speed automaticity on the part of the user may be frustrated by a seemingly inconsistent system. A second problem is that the menu utilization profile may not be just user specific but may also vary drastically depending on the task… Finally, there is yet no empirical research supporting the idea of self-adapting menus and their superiority over user-controlled rapid access methods. [p. 251]

The rapid access methods Norman is referring to include the use of hotkeys (e.g., Ctrl-S for "Save") which allow the user to effectively select a function without having to scan through the menu.

Contrary to Norman's assertion, there have been at least two instances of adaptive menus that have been shown to increase user performance. These are the Adaptive Telephone Directory by Greenberg and Witten [1985] and Split Menus by Sears and Shneiderman [1994]. These menus will be described in the Adaptive User Interfaces (AUIs) section later in this chapter.

11) Norman cautions that designing systems that allow users to customize or build their own menu structures will place a burden on the user to become a designer which will then require considerable guidance to avoid irrational design.

The idea of restructuring the menus in MSWord (i.e., re-ordering the menu items so that related items are better spatially located) is a design idea that we considered while engaged with our Study One. The problem with this is that despite users' complaints about the menu structure, they have come to know the structure regardless of its design merits.

A number of advances in menu design have taken place since Norman's book was published. Two recent non-adaptive designs are highlighted next.

A marking menu is a specialized radial (or pie) menu; it can be operated in a standard fashion whereby the menu pops up and the user selects the desired menu item (represented by a pie section), however, the user is also able to perform the gesture of selection without the menu popping up [Kurtenbach & Buxton, 1994]. The gesture simply involves making a straight mark in the direction of the desired menu item. The primary goal of marking menus is to make menu selection quicker and to make the transition between novice and expert behaviour seamless. A longitudinal evaluation with 2 subjects found that marking was significantly faster than selection with the menu present and that a user's skill with marking menus increased with use. Users began by using the menu but with practice would gradually switch over to making marks. After a period of non-use users would fall back to relying on the menu but would reacquire the skills to make marks once again. Refinements on the marking menu design are given by Tapia and Kurtenbach [1995].

Hotbox is a new interface widget that combines several GUI techniques that are usually used independently, including accelerator keys, modal dialogs, pop-up/pull-down menus, radial menus, marking menus, and menubars [Kurtenbach, Fitzmaurice, Owen, & Baudel, 1999]. Hotbox was designed for a professional 3-D computer animation application and the goal was to make all of the approximately 1200 commands accessible, to support fast access to the frequently used commands, and to unify novice and expert behaviours. Informal feedback from beta customers appeared to be promising.

The Hotbox is an interesting design, however, the implicit assumption seems to be that all novices will eventually become experts users of the application. In the case of a highly specialized application this assumption may be quite valid. On the other hand, for applications in which users can be productive with only a small subset of the commands, the hotbox widget is not likely to be an appropriate design.

2.4.3 Customization/Personalization

Most software systems (both operating systems and applications) offer forms of customization that allow users to manage functionality in a way that might better suit their needs. There has been little reported in the literature about how users customize GUI applications, however, a few studies that have investigated this topic are summarized below. To avoid any confusion we first clarify the terminology.

Customization with respect to interfaces is a term that has traditionally been used to describe a change or changes that have been made to the default interface and/or its behaviour. Customizations are usually changes that are made by the user him/herself or by someone on behalf of the user, such as a system administrator (who may have made the same changes for all users in a particular group). In the late 1990s *personalization* became somewhat of a buzzword although it has never been clearly defined [Riecken, 2000]. It tends to refer to changes to the interface that are inherently personal and that are either made by the user him/herself (which is the same as customization done by the user) or are carried out by the system itself; artificial intelligence of some form adapts the interface based on what it knows about the user (e.g., the user's past performance). The latter form of automatic personalization has been called "self-customizing software" by some people [Schlimmer & Hermens cited in Hirsh, Basu & Davison, 2000]. Personalization achieved through artificial intelligence versus that achieved through user customization differs predominantly in the amount of control the user has over the changes to the interface.

In this section we discuss customization by the user. Interfaces that rely on artificial intelligence will be discussed in Section 2.4.4.

Page, Johnsgard, Albert, and Allen [1996] conducted a field study of 101 users of the commercial word processor Word Perfect 6.0 for Windows to determine the extent of customization undertaken. Participants' use of the software was monitored for 28 days and their usage and customization settings were collected. Contrary to their expectations, they found that most users did customize. In fact, a surprising 92% of the participants did some form of customization to their software. One explanation for this finding, however, is that Page et al. defined customization very broadly. For example, changing the zoom setting to something other than the default 100% was considered a customization. In addition, they did not distinguish whether it was the participant him/herself who had actually undertaken the customization or whether someone else had made the change on the participant's behalf. Some companies create custom elements (e.g., macros or toolbars) for all employees. While this is still a form of customization, it should be identified separately from user customization.

A relatively new form of customization is seen on the World Wide Web, that of personalized web pages for web portal sites. One example is Yahoo!, which provides a large information repository that can be both browsed and searched by the user[8]. Users have the option to set up a personal page (My Yahoo!) that contains information of interest and then use this page as their entry to the Yahoo! repository rather than the main entry page. The effectiveness of My Yahoo! was described in a recent article by Manber, Patel and Robison [2000]. They indicated that the majority of active My Yahoo! users did not customize their pages but rather worked with the default personal page. Three possible reasons were given for the lack of personalization: the default page was good enough, the customization tools were too difficult to use, or most people did not need complex personalization. Manber et al. expected that the real reason was a combination of all three. In their experience, they found that people generally did not understand the concept of personalization and that personalized features were of greatest benefit to power users. They noted that a major challenge for personalization is to make it accessible to less-experienced users. To do so it must be easier to customize pages and it must be made clear to novice users that customization is a possibility.

Mackay studied patterns of software customization in the Project Athena Unix/X Window System [1990, 1991]. She collected customization files[9] and interviewed 51 users over a 4-

[8] Yahoo! [2001] is one of many portal sites on the World Wide Web.

[9] In our Study One we collected information about our participants' MSWord customization by taking screen captures from their computers while the MSWord application was displayed on the desktop.

month period. Unix customization files can be extremely complex and Mackay found that only a small number of highly skilled users found the customizability to be useful; most other users who were less skilled and also less interested in the software found these files to be overwhelming. Interestingly, the senior technical users tended to customize very little because they were no longer interested in "playing" with systems. Mackay [1991] identified a group of users who were skilled but not among the highly skilled who were much more interested in interpreting the needs of their colleagues and creating customization files tailored to those needs. She named them *translators* because they translated between the highly technical group and the rest of the organization. Others have identified this role and assigned their own names: *tinkerer* [MacLean, Carter, Lovstrand, & Moran, 1990], and *local developer* [Gantt & Nardi, 1992].

Mackay [1991] found that there were triggers and barriers to customization and they could be grouped into the following categories: external events, social pressure, software changes, and internal factors. Examples of triggers included: a job change, a colleague encouraging use of his/her personal customization, system upgrades, and being fed up with the inefficiency of repeating the same sequence of steps. The most common barriers to customization were lack of time, difficulty in modifying settings, and lack of interest. Mackay found that users tended to rely heavily on their customizations and therefore strongly resisted new software releases that required behaviour changes on their part.

Based on her qualitative investigation, Mackay [1990, 1991] identified the time tradeoff involved in customizing. By sharing customizations users can spend less time learning how to customize (and potentially making errors) and thus more time accomplishing actual work and can also learn how others work and thus benefit from the innovations of others. Mackay used an economic analogy: when a user takes time to customize the user is trading off a short-term investment of time for a longer-term potential gain in productivity.

To summarize, unless we define customization broadly, it was found that the majority of users tend not to customize or customize very little, and they prefer to make use of customizations that others have made, rather than discover their own customizations. One of the main reasons for this appears to be the time required to learn and use the customization facility. Research into the issue of how to make customization facilities more usable, however, is largely absent from the literature.

2.4.4 Intelligent User Interfaces (IUIs)

We turn finally to design solutions that fall under the broad category of intelligent user interfaces, the general goal of which is to assist the user by having the machine offload some of the complexity from the user [Miller, Sullivan, & Tyler, 1991]. Maybury and Wahlster [1999] define IUIs to be "human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media" [p. 2]. Among other things[10], this includes dynamically reconfiguring the interface to accommodate individual differences (adaptive user interfaces – AUIs) and carrying out mundane tasks that the user would rather not do him/herself (intelligent agents). In order for a system to self-adapt or to suggest adaptations that are appropriate to the user, the system must have knowledge about the user. This knowledge is generally maintained in a knowledge base called a user model, which is responsible for acquiring and managing data as well as providing a means for the applications (or consumer of the user model) to access the data. Whether one actually thinks of an interface as "intelligent" in the true meaning of the word depends on the sophistication of the user model and how the interface changes in relation to the user model.

We review agent and AUI literature separately below, provide a few examples of each, and then briefly summarize our review. The examples have been selected based on how closely they relate to the goals of our research.

Agents

There has been considerable controversy over agents, the foremost contention perhaps being the lack of a clear definition. Highlighting some of the issues raised in this debate, which has

[10] IUI research includes topics such as multimodal input and automated output generation (e.g., automated graphics design). These are of relatively lesser relevance to the research described in this dissertation and the reader is therefore encouraged to see the text on IUIs by Maybury and Wahlster [1999] for further information.

centered on the value of using agent technology in the interface rather than more established techniques such as direct manipulation, serves as a reasonable introduction to agents. In two recent debates between Pattie Maes, a leading researcher in the area of agent technologies, and Ben Shneiderman, a long time proponent of direct manipulation, it seems that some form of consensus may be on the horizon [Shneiderman & Maes, 1997], which we summarize in this section.

Maes acknowledged that the word *agent* is overloaded. The area of agent research that relates to complex technology is what she calls *software agents.* She specifically prefers avoiding the terms *intelligent agents* and *autonomous agents* because these terms are problematic. Maes gave the following advantages of software agents:

- A software agent knows the individual user's habits, preferences, and interests.
- A software agent is proactive. It can take initiative because it knows what the user's interests are. It can, for example, tell the user about something that he/she may want to know about based on the fact that he/she has particular interests. Current software is not at all proactive. All of the initiative has to come from the user.
- Software agents are long-lived. They keep running, and they can run autonomously while the user goes about doing other things.
- Software agents are adaptive in that they track the user's interests as they change over time.

An agent can act on the user's behalf while he/she is doing other things in much the same way that a travel agent will act on an individual's behalf once his/her travel needs and preferences have been made known. One example by Kozierok and Maes [1993] was an agent that learned the user's needs and preferences with respect to scheduling meetings by observing the user, through reinforcement (direct feedback from the user), and by direct instructions from the user.

Maes noted the following reasons that we need software agents today:

- Our computing environment is no longer closed and under a user's complete control like it once was. Our computer provides a viewport into a vast and dynamic network of information and other people.

- The typical user is no longer a computer professional.

- People use their computer for more and more tasks and are thus required to keep track of more and more information.

Shneiderman [1995] has argued in the past that "the effective paradigm for now and the future is comprehensible, predictable, and controllable interfaces that give users the sense of power, mastery, control, and accomplishment" and that the term agent itself is ill-defined but seems to include the following components:

- anthropomorphic presentation
- adaptive behaviour
- accepts vague goal specification
- gives you just what you need
- works while you don't
- works where you aren't

He notes that the first three seem appealing at first, but have proven to be counterproductive. The latter three are good ideas but can be achieved more effectively with other interface mechanisms.

Despite some of Maes' earlier research directions with agents, she has stated that it is a misconception to think that agents are necessarily personified or anthropomorphized. In fact, she noted that most agents are not. Maes also clarified that agents do not necessarily rely on traditional AI techniques, like knowledge representation and inferencing. Many of the commercially available and successful agents rely on either user programming or machine learning rather than traditional AI techniques.

Maes argued that agents are not an alternative for direct manipulation, but rather they are complementary metaphors: an agent is not a substitute for a good interface. She argued that the reason for agents is delegation and that no matter how good the interface, there are some tasks that she just may not want to do herself. She gave the example that if her car had a perfect interface for fixing the engine, she still would not fix it.

Maes also addressed the criticism that agents make the user dumb and that they usurp all control from the user. Concerns about agents are addressed by the guidelines for agents presented by Maes at the 1997 International Conference on Intelligent User Interfaces [Computer Science and Telecommunications Board, 1997]:

1) Make the user model available (inspectable, modifiable) to the user.
2) The agent's method of operation should be understandable to the user.
3) The agent should be able to explain its behavior to the user.
4) The agent should have the ability to give continuous feedback to the user about its state, actions, and learning.
5) The agent should allow variable degrees of autonomy, and the user should decide how much and what type of tasks to delegate to the agent. The user should be able to "program" the agent (e.g., teach it things, make it forget things).
6) The user should not have to learn a new language to deal with the agent. One goal is to use the application to communicate between the agent and the user.

These guidelines serve to move the agent research much closer to Shneiderman's requirements for controllable and predictable user interfaces. Furthermore, Maes noted at the second of the two debates that one of the key reasons for the division in philosophies is that the two camps are focusing on different problem domains. The Shneiderman camp is dealing with a well-structured task domain and a well-organized information domain, one that lends itself to visualizing all of the different dimensions, for example, visualizing the information stored in a database. The Maes camp, on the other hand, is dealing with an information domain that may be very ill structured and very dynamic. An example of such a domain is the World Wide Web.

Shneiderman [1995] cited a number of examples where consumers have continually rejected anthropomorphic terms and concepts and, in fact, Maes noted that the most successful software agents thus far are ones that are pretty much invisible. Despite this, there are at least a few examples where agents are visible and are used to help the user navigate the interface itself. These agents do not adhere to the delegation philosophy of agents and demonstrate that research on agents is diverse.

One well-known example of a visible agent that has appeared in commercial software is the Microsoft Office Assistant, also known as Clippit, the anthropomorphic paperclip. This character literally “sits” on the screen and tries to offer helpful advice to the user and accepts natural language queries. Clippit was first introduced in MSOffice 97 and grew out of the Lumière Project that investigated Bayesian user modeling for inferring the goals and needs of software users [Horvitz, Breese, Heckerman, Hovel & Rommelse, 1998]. Goals are defined as the target tasks or subtask at the focus of a user’s attention. Needs are information or automated actions that will reduce the time or effort required to achieve the goals. Although no user evaluation of this agent has ever been reported in the literature, the popular press has been very unfavourable. In fact Microsoft “retired” Clippit in the MSOffice XP version released in the year 2001 and the following excerpt from an online publication was representative of the response in the popular press:

> Microsoft is retiring Clippit, the annoying Office help character everyone loves to hate. If you’ve never used Word or other programs in the Office suite, you probably haven’t gotten a chance to know just how annoying Clippit and his Office Assistant friends can be. How can you like someone who butts in while you’re trying to get something done to say intelligent things like, “I see you’re trying to write a letter ...” [I see you’re going to retire, Clippit, 2001]

Those who worked on the Lumière Project noted that the Bayesian user modeling that was actually implemented in the MSOffice software was far less sophisticated than the modeling that was originally envisioned. For example, Clippit used broader but shallower models, reasoning up to thousands of user goals in each Office applications, rather than using rich combinations of events over time, and the system only considered a small set of relatively atomic user actions [Horvitz et al., 1998]. No reason was provided as to why the more sophisticated modeling was not used.

A second example of a visible agent by Rich and Sidner [1996] is called a collaborative interface agent. This agent mimics the relationships that exist when two humans collaborate on a task involving a shared artifact such as two mechanics working on a car engine together or two computer users working on a spreadsheet together. There was no user testing documented in this research.

There exists non-anthropomorphic agent technology that is relevant to our research, namely, the work on a commercial word processor carried out by Linton et al. [1999, 2000]. They proposed a *recommender system* [Resnick & Varian, 1997] that alerts users to functionality currently being used by co-workers doing similar tasks. The design incorporates collaborative-filtering technology that determines which functions would be good candidates for a user to next learn and introduce into his/her command repertoire. These commands are then recommended to the user through a tip-of-the-day like facility. The user controls when to view a tip and can easily ignore the tip. No user testing of this design was reported.

Dryer [1997] provided a brief discussion on *wizards* and *guides* which he claims to be the most common kind of UI agent today. The goal of wizards is to assist the user by breaking a complex task into a series of steps and then presenting one step at a time to the user. Wizards work best with a linear series of steps and are therefore most successful when the tasks have algorithmically derived solutions. Dryer noted that these agents generally do not use any artificial intelligence although they are often perceived by users to be intelligent. The benefits of a well-designed wizard are that a multi-purpose task interface is replaced by a task specific interface that guides the user along an efficient path to task completion and that autonomously completes those steps of the task that do not require the user's attention. One possible disadvantage of such a constrained process is that the user may not actually reflect on his/her actions and therefore may not learn from the process.

Based on Dryer's [1997] account, it wasn't entirely clear what guides are. He provided the following description: "Guides are another kind of UI agent. Typically, guides provide task assistance by monitoring a person's interaction with the information system and presenting information appropriately. ... Guides are intelligent because they annotate an interface whenever and however it is most likely to be useful. A well designed guide will direct a person through the next step in a task." According to Dryer [1997] guides are best for frequent tasks because users want to learn about tasks they do frequently and wizards are best for infrequent tasks that users don't necessarily want to learn about but do need to accomplish. He further suggests that wizards work best for solutions that can be derived algorithmically whereas guides can assist either algorithmic tasks or heuristic tasks.

Myers, Potosnak, Wolf, and Graham [1993] discussed the use of heuristics[11] to predict users' intentions. They noted that systems that use heuristics attempt to delegate some of the low level details in order to save the user time and are also hopefully easier to learn because the system does part of the work. The disadvantages are similar to those already mentioned for agents: an incorrect action could occur which might not be noticed by the user; the user might not understand why the system did a different action or how to get the desired action to occur; the user might have the feeling that the system was unpredictable and that he/she no longer had control; there might be additional user testing costs to determine whether the heuristics are sufficiently predictable to users and to tune the heuristics or their presentations; and there might be additional documentation and quality assurance costs. Myers et al. suggested that successful use of a heuristic requires that:

1) A majority of users would predict the same result of an action performed in a given context.

2) An algorithm can be developed which interprets the context and produces the result most users expect.

[11] Heuristics are problem-solving techniques in which the most appropriate solution is chosen using rules [Myers et al., 1993].

3) In cases where the algorithm does not do what a user requires, it should still give a result that users interpret as reasonable, and the result must not be harmful.

4) It should be reversible through some type of "undo" facility.

5) The user can discover ways to override the default behaviour when necessary.

These guidelines for heuristics and Maes's guidelines for agents show the importance of user expectation and control when some tasks are delegated to the system.

Adaptive User Interfaces (AUIs)

Research in adaptive user interfaces addresses the diversity of the user population differently than agent research. Rather than delegating cumbersome tasks to a trustworthy agent or collaborating with an agent in order to navigate the interface, the philosophy of AUI research is that the graphical interface should adapt to the needs, preferences, and skills of the user. The goal is for the complexity of an adapted interface to be less than that of an equivalent all-in-one interface because the functionality accessible matches the user's needs, preferences, and skills. Cote-Muñoz [1993] noted the following expected benefits of AUIs: the user will better master the complexity of software, there will be better user performance, the system will be able to gain and maintain the attention of the user, and the system will avoid overloading or "underloading" the user.

The question of who adapts the interface is relevant. Tyler and Treu [1989] identified and discussed three possible sources of adaptation: a computer expert, the user, or the system. In the first case, a computer expert, perhaps the original designer of the system modifies the interface based on user feedback. The problem with this scenario is that it doesn't scale; an expert may be able to construct a few differently customized interfaces, however, this will not likely be sufficient to meet the needs of all the different users. Another problem that was not mentioned by Tyler and Treu is the substantial delay or turnaround time generally required for modifications. The second possibility is for the user to take advantage of the customizability that most systems provide by tailoring the interface to suit his/her own style and abilities. This scenario has already been covered in some depth in Section 2.4.3. It works well for experienced users who know both what needs to be tailored and how to go about tailoring. However, it doesn't hold for inexperienced users who are not likely to know what they need to customize and how they should go about customizing [Innocent, 1982; Page et al., 1996]. The third approach considered by Tyler and Treu is that the system adjusts the features of the interface based on acquired knowledge about the individual user. The basic idea is that the system monitors the user and adjusts the interface to suit that individual.

Given that the goal of adaptive interface research is to match the interface to an individual's profile, the solution of having a computer expert in charge of adapting the interface is not practical. This leaves responsibility for adaptation to the user or to the system itself. Dieterich, Malinowski, Kühme, and Schneider-Hufschmidt [1993] provided a survey of the AUI literature and a framework for understanding the adaptation process. They identified four stages in the process of adapting a user interface:

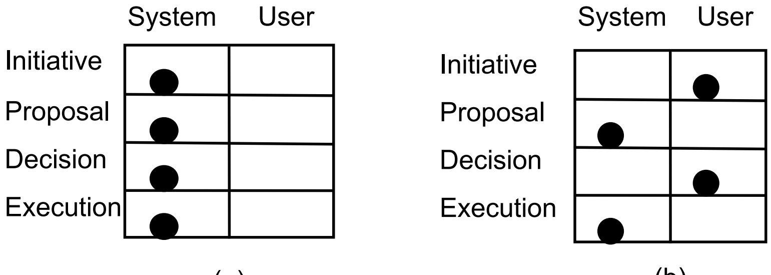
1) *Initiative*: the need for adaptation is suggested
2) *Proposal*: alternatives for the adaptation are proposed
3) *Decision*: one of the alternatives is chosen
4) *Execution*: the chosen alternative is executed

At each of the four stages it is either the user or the system that is in control. For example, when the system controls each stage, the system is essentially a self-adaptive system. When the user is in control of all stages, the user is doing the adaptation, which is tantamount to customization. These two extremes are sometimes juxtaposed as *adaptive* (full system control) vs. *adaptable* (full user control) systems [Fischer, 1993]. Then of course there are different control combinations. Figure 2-2(a) represents the configuration in which the software is entirely self-adaptive: the system is completely in control. Dieterich et al. [1993] noted that the majority of research into AUIs has been from the system-adaptive perspective and they make the following conclusions from their survey:

- Systems that have user control seem more promising than those that have all/mostly system control. A configuration in which the user and the system share control, which they call Computer-Aided Adaptation, is deemed to be the most promising approach in order to obtain a user interface that will help the user to perform his tasks in a pleasant and effective way. This configuration is depicted in Figure 2-2(b).

- More effort should be spent on the integration of adaptivity mechanisms into common user interface design and management tools.

- Aspects of user acceptance and the evaluation of adaptation have been neglected in the past and need far more attention.



	System	User
Initiative	●	
Proposal	●	
Decision	●	
Execution	●	

(a)

	System	User
Initiative		●
Proposal	●	
Decision		●
Execution	●	

(b)

Figure 2-2: Categorization of adaptive systems: a) Self-Adaptive b) Computer Aided Adaptation.

The first point given above advocates user control and can be seen to parallel the more recent direction in agent research that acknowledges the importance of user control and user understanding of the agent behaviours.

To achieve the Computer-Aided Adaptation configuration, Kühme [1993] proposed an inspectable user model which gives the user an insight into adaptation strategies and underlying assumptions. The user should clearly be made aware of the existence of the user model and should have access to the included information. By changing the information in the user model, the user is essentially adapting the interface in an implicit manner. (See Csinger, Booth, and Poole [1995] for another example of an inspectable user model.) The user should also be able to adapt the interface explicitly by inspecting and adjusting the adaptation-related mechanisms. Kühme proposed an adaptable dialog monitor that collects information relevant for adaptation and an adapter, which adapts the dialog by applying the information represented in the user model.

Greenberg and Witten [1985] conducted some early work on adaptive interfaces and noted some advantages and disadvantages to adaptation. The advantages are:

1) variations in expertise across users
2) evolving user needs
3) user has appropriate control
4) attempts to rectify user-designer conflicts

The disadvantages are:

1) dynamics of user-system concurrent modeling (at the same time the adaptive system is trying to make a model of the user, the user is trying to model the system)
2) user will lack confidence in a system that seems inconsistent (although Grudin [1989] argues successfully that consistency is only one of many competing design goals)
3) user does not have appropriate control
4) complexity of implementation
5) inaccuracies of model construction
6) difficulties in evaluating adaptive systems

From the above we can see that control, which appears both as an advantage and a disadvantage, is indeed controversial and furthermore it is not clear what constitutes appropriate control.

Stephanidis, Karagiannidis, and Koumpis [1997] document a methodological approach to adaptive systems. They note that, in most adaptive systems, the adaptation strategy is hard-coded into the system, and therefore, when changes are needed, it is relatively difficult to implement the changes. They focus on the adaptation strategy as a decision making process, which is characterized by the following attributes:

1) *What* to adapt: aspects of the user-computer interface that are subject to adaptations are called *adaptation constituents* and can be semantic, syntactic, or lexical;

2) *When* to adapt: aspects of the interaction called *adaptation determinants* on which the adaptation decisions are made;

3) *Why* to adapt: the *adaptation goals* underlying the adaptation process;

4) *How* to adapt: adaptations are driven by a set of *adaptation rules* that essentially assign certain adaptation constituents to specific adaptation determinants for given adaptation goals.

The approach given by Stephanidis et al. [1997] enables:

- the customization of the set of adaptation determinants and constituents;

- the incorporation of the adaptation goals as an integral part of the adaptivity process;

- and the modification of adaptation rules, according to the goals of adaptivity.

We next describe several examples of adaptive user interfaces.

Adaptive Telephone Directory

Greenberg and Witten [1985] provided empirical evidence that an adaptive user interface can indeed improve user performance. They designed two different versions of a menu-driven telephone directory. One version had a straightforward interface where the menu structure was pre-computed and remained static across all uses. In the second version the menu structure adapted to the user's interactions such that the most frequently accessed names were located at the upper levels of the hierarchy and the least frequently accessed names were located at the lower levels of the hierarchy. This design reduced the number of menus that needed to be traversed in order to access the popular names.

A study that included 26 subjects was conducted to compare the adaptive directory to the static directory and the results strongly favoured the adaptive directory. The task of locating a given person was 35% faster and the error rate was 60% lower with the adaptive directory.

Greenberg and Witten [1985] noted that their adaptive directory was a very specific and highly constrained example of an adaptive interface. They cautioned against over generalizing their findings, but rather suggested that their empirical results should be seen as an existence proof that an adaptive interface can be beneficial.

Split Menus

Split menus, created by Sears and Shneiderman [1994], are a form of adaptive menu in which the menu is split into two sections. The top section includes the frequently used menu items and the bottom section includes the infrequently used menu items. It is expected that split menus will be beneficial in situations where a small subset of the menu items represent the majority of selections. In addition, the potential benefits of split menus are expected to increase as the length of the menu increases.

Sears and Shneiderman [1994] conducted two field-style usability evaluations of split menus at two separate sites as well as one laboratory study. The usability studies evaluated performance times on relatively long font menus (average number of menu items at one site was 11, and there was a fixed menu of 28 font items at the second site) and results showed that selection times were reduced by 14% to 58% depending on the field site and the menus. The controlled experiment compared the performance of split menus to alphabetically ordered menus and frequency ordered menus. The later refers to a dynamic menu where the top-most menu item is always the most frequently used, the second item form the top is second most frequently used, etc. All comparisons included menus of a fixed-length, namely 15 items, and there were always 3 frequently used items of the 15 items. Results showed that when the 3 frequently used items were located in the *bottom* third portion of the alphabetic menu that both the split menu and the frequency menu were significantly faster than the alphabetic menu. On the other hand, when the 3 frequently used items were located in the *top* third portion of the alphabetic menu, then both the split menu and the alphabetic menu were significantly faster than the frequency menu. Split menus resulted in significantly higher subjective preferences compared to both the other menu alternatives.

Adaptive Prompting

Malinowski, Kühme, Dieterich, and Schneider-Hufschmidt [1993] introduced the idea of adaptive prompting. This technique aimed to reduce the user's confusion caused by the volume of prompts (menu items, dialog boxes, etc.) by leading the user to the most relevant functionality in a given situation. Adaptive prompting was provided to the user through a complementary pre-selection of the relevant options and was not a substitute for existing interaction techniques.

Malinowski et al. [1993] discussed two different adaptive prompters. The first was the Adaptive Action Prompter, which was a permanently visible, dynamic menu (or control panel) that included only the most appropriate and most likely to be chosen actions based on the user's context. Thus the prompter contents were updated with every context change. The user could always select actions from either the prompter or the regular menus, whatever was more convenient in a given situation. Because the prompter listed the most appropriate actions in one place the user had a good survey of sensible alternatives. The prompter could be shown as plain menu items alone, or menu items that indicated the referred object (e.g., "CUT selected text" where selected text is the referred object), or menu items with task-oriented explanation (e.g., "START to start the simulation using the selected sample"). The user could optionally be involved in controlling the rules used for the prompting.

The second form of adaptive prompting discussed by Malinowski et al. [1993] is Adaptive Dialog Boxes. This prompting was to address the problem that dialog boxes often present a lot of parameters that are required for a function. They suggested that strategies for coping with the volume of parameters such as sorting the parameters based on frequency of use or moving rarely needed parameters to an additional dialog box titled something like More Parameters[12] changes the layout of the dialog box and therefore results in confusion to the

[12] This solution of having a button in a dialog box that permits access to more advanced options appeared in the Xerox Star. The term "Progressive Disclosure" has been used by the original Star design team to describe this model [Johnson et al., 1989].

user. The strategy of adaptive prompting in dialog boxes was to present the information in a way that allows the user to identify the important items and their parameter settings at a glance. This was achieved by using forms of highlighting and color-coding to draw the attention of the user. The structure of the dialog box was not changed in this approach. No user testing for either type of adaptive prompters was documented.

AIDA

Cote-Muñoz [1993] documented an adaptive system for interactive drafting and CAD applications that fixed the amount of functionality offered by icons and menus based on the user's knowledge. To support exploratory learning, however, the user had access to the full system's functionality through a command line. In addition, new functionality was introduced in one of two ways: either the user created a new command/macro or the system recognized a repetitive operation (such as creating three lines that resulted in a triangle) and suggested the creation of a command/macro. These command/macros then became available to the user through new menu items. No user testing was documented.

Skill Adaptive Interface

According to conventional wisdom, direct manipulation interfaces are better for novices and command line interfaces are better for experts. Some attempts to create hybrid applications (applications that include both direct manipulation and command line interfaces) have been made. Gong and Salvendy [1995] studied hybrid systems and found that most users never moved beyond using the menu. To rectify this they created the Skill Adaptive Interface (SAI) that pushed the user to use the command line. Once a user had selected a menu item a threshold number of times, all subsequent times the same item was selected the system would provide a prompt that gave the equivalent command. The user was then forced to use the command line for this item. Gong and Salvendy reported user studies that showed this hybrid technique to be promising.

In addition to forcing a user into command line usage, SAI was also adaptive in that the menu contents were variable. Once the user was able to enter the command at the command line without using the prompt available from the menu, the menu item was transferred from the active menu to a hidden menu that was attached at the end of the active menu. Users had access to the hidden menu, although Gong and Salvendy did not describe this access [1995]. The hidden menu contained items for which the user had already learned the corresponding command and also items that the user had not yet encountered. When an active menu item was transferred to the hidden menu, an empty menu slot was then available. To fill the empty slot, the system selected an item from the hidden menu that had not yet been made visible. The determination of which item should be selected for the empty slot was based on a priority parameter. Gong and Salvendy do not elaborate any further on how this parameter was set; there was no user testing of the adaptive aspect of the system.

Flexcel

Thomas and Krogsœter [1993] noted that the great majority of research in adaptive systems had been done with prototype systems. Given the limitations of doing research on adaptation with prototype systems, they had the explicit goal to assess adaptation in a complex software product that was commercially used [Krogsœter, Oppermann, & Thomas, 1994]. Their product of choice was Microsoft Excel because of their ability to modify the Excel interface through the Excel dialog editor and the macro programming language.

In general, the adaptations that were supported by Flexcel appeared to focus on streamlining the number of steps to execute a desired operation. For example, if a user often printed documents in such a way that required specifying the same attributes each time (e.g., number of copies, page range, and duplex/simplex), a one-click access to execute this print operation could be defined. More generally, the following adaptation features were implemented in Flexcel:

1) The user could define new menu entries and new key shortcuts for function parameterization.

2) The user could define key shortcuts for Excel functions, which normally can only be invoked from the menu.

3) For functions with default parameters, the defaults could be changed.

4) A separate adaptation toolbar was made available to make the aforementioned tools more visible.

5) An acoustic signal and a blinking button indicated system-generated adaptation suggestions. The user could access the suggestions at his/her convenience. Unread suggestions were maintained in a tip list.

6) Usage suggestions reminding the user of seemingly forgotten adaptations and a critique model telling the user how the adaptation tools may be used more efficiently were also included.

The design of Flexcel was clearly attempting to assist the user without taking over control entirely. The first Flexcel prototype was evaluated with 5 users and the second with 8 users, both in one-session informal laboratory studies. Among other things, Krogsœter et al. [1994] found that the transition between the user accepting system-defined adaptation suggestions to actually initiating adaptations him/herself was not satisfactory, that it was impossible to discern adaptations that would exactly meet the users needs based only on observing what the user had done before, and that users had difficulty coming up with meaningful menu labels for newly added menu items.

Adaptive Menus in Microsoft Office 2000

The MSOffice 2000 software package which includes MSWord and other productivity applications made a significant departure in its user interface from that in the Office 97 package by offering adaptive menus and toolbars [Microsoft Office 2000, 2000]. When a menu is initially opened, the default is to display only a subset of the menu contents (approximately a third to a half of the menu items). By hovering in the menu for a short period (1 or 2 seconds) or by clicking on a “double-down arrow” icon at the bottom of the menu the “long” menu can be displayed. After selecting an item from the “long” menu, the next time the menu is invoked the selected item will appear in the “short” menu, but after some period of non-use, menu items will disappear from the “short” menu but will always be available in the “long” menu. There are clear parallels between the concept of a short menu and the concept of a working set found in system memory management. (See Appendix A for some screen captures of this interface.) Users do not have access to the underlying user model; there is no explanation of which data are collected, how the adaptive algorithm works in conjunction with the data, and users cannot alter the algorithm. The only control available is to turn the adaptive menus on/off and to reset the data collected in the user model. No user testing of this interface has been reported in the literature.

History Mechanisms

As does history, users tend to repeat themselves. Described below are two mechanisms that capture the user's interactions and facilitate their repetition.

Based on his study of command usage in UNIX (described in Section 2.3.2), Greenberg proposed Workbench as a design solution for handling the complexity of command-based systems [1993]. He labeled systems in which users use only a small subset of the command repertory as *recurrent systems* and he created a reuse facility – a front end that collects the user's commands and then makes them easily accessible for reuse. Workbench is a graphical reuse facility.

A web browser is another instance of a recurrent system in that the majority of the pages seen by a user are in fact pages that the user has visited previously. Tauscher and Greenberg [1997] identified that users revisit 58% of the pages they see. Kaasten and Greenberg [2001] noted that there are a number of mechanisms in all browsers that assist the user in revisiting pages. These include the Back button, the History list, and Bookmarks. However, each of these mechanisms has limitations: the Back button is stack-based rather than recency-based, the History list often provides an extensive set that must be scanned by the user, and Bookmarks must be explicitly maintained, which is an added task for the user. Perhaps the largest problem is that there is no integration between these three mechanisms. Kaasten and Greenberg proposed a revisitation system that integrated the three previously mentioned mechanisms. It provided a recency-ordered list with duplicates removed. To facilitate scanning, a thumbnail image was included with the title of the web page. The Back button was altered to operate on the recency list and bookmarking was incorporated by enabling the user to explicitly mark pages within the list. An integrated search mechanism was also provided.

No user testing was reported on either the Workbench or the revisitation system. Given that a recency list is a relatively predictable model compared to some of the other adaptive models described in this chapter, one might expect it to be more easily adopted by the user.

IUI Summary

Our review of the Intelligent User Interface literature as a whole has shown that although it is certainly motivated by a desire to assist the user, it has largely been technology driven. There has been very little user testing conducted [Maybury & Wahlster, 1999], which is a substantial deficiency with this body of research. This can partially be explained by the fact that evaluation of IUIs is more complex than that of standard interfaces – there is greater variability with IUIs and the evaluation methodology needs to accommodate this variability [Maybury & Wahlster]. There does appear, however, to be recognition that evaluating real systems with real users is necessary: “More practical systems must be developed and evaluated for further progress.” [p. 420, Cawsey cited in Maybury & Wahlster].

Perhaps the largest issue that needs to be addressed is that of user control. The IUI community has certainly acknowledged that some user control is required but understanding the nature of the boundary/balance between user and system control is still much in need of research. Greenberg and Witten’s [1985] early work on the adaptive telephone directory and Sears and Shneiderman’s [1994] split menus both provide empirical evidence that users can relinquish some control and still benefit from a straightforward adaptive interface. More research that compares adaptive to static interfaces needs to be carried out in order to determine more precisely where the benefits and pitfalls of adaptive interfaces lie.

2.4.5 Summary

A number of design approaches for complex software have been presented. Functionality blocking and the Training Wheels interface in particular have shown promise but a real design solution needs to accommodate all levels of expertise and varied tasks. Allowing users to customize their interfaces certainly puts control in the hands of the user, however, there has been little evaluation of customization facilities and the evaluation that has been done has shown that they are too sophisticated for many users. This prompts one to ask whether or not these customization facilities provide real control or only the semblance of control. Design approaches that rely on artificial intelligence have been largely technology driven rather than user driven and have therefore predominantly resulted in designs that have suffered from a lack of user control. There has been some success in simple adaptive interfaces but further work in this area is required.

To our knowledge there has not been a study that has captured functionality usage and users' overall experience of a complex application. Certainly there has not been a design solution to complex software that has been derived from such a comprehensive study. The research described in the remaining chapters of this dissertation is intended to provide the first step in this direction.

CHAPTER 3 Study One – Capturing Users' Experience of Complex Software

The purpose of Study One was to fulfill our first research objective (given in Section 1.2), namely, to gain a more systematic understanding of users' experiences with complex software. The content of this chapter is the result of a collaborative effort with Dr. Gale Moore, a sociologist at The University of Toronto, and draws largely from the published paper "Are We All in the Same 'Bloat'?" by McGrenere and Moore [2000]. This study, here referred to as Study One, fits into a broader research project called *Learning Complex Software*, of which Moore is the project leader.

As previously mentioned, there has been considerable interest in the past few years in what has been termed "bloat", "bloatware", and "creeping featurism". One implication from these labels has been that a bloated application is one in which there are a large number of unused features. In the popular press, "bloat" has often been used as a catch-all term to suggest, negatively, that an application is filled with unnecessary features. But, do users actually *experience* complex software applications in this way? Do people feel overwhelmed by the number and variety of choices in the interface, and if they do, how do they handle this? Our study of 53 Microsoft Word, Office 97 users from the general population provided an opportunity to explore these questions in detail.

First, we took a relatively straightforward quantitative approach, counting functions and defining software as "bloated" if a significant proportion of the functions available were not used by the majority of users. Second, we used qualitative methods to ground our findings by placing them in the context of the users' narrative reports and questionnaire responses. Third, we evaluated and extended a study based on work done by Microsoft [Kaufman & Weed, 1998a, 1998b] which distinguished two profiles of users according to their perception of "bloat". Thus we had several distinct methodological approaches, each offering a unique perspective on the research problem. By triangulation of methods[13] we were able to more fully understand the user's experience of complex software applications, and to gain insights that could be applied to interface design.

Study One received ethics approval for the use of human subjects from the Social Sciences and Humanities Review Committee at the University of Toronto.

3.1 Study Design

3.1.1 Software Application Studied

The choice of a specific implementation of a word processor allowed us to control for one potential source of variation. Microsoft Word, Office 97, running on the PC (MSWord) was selected. We used the number of functions available in an application as our indicator of complexity[14]. Because our primary question had to do with the user's perception of complexity, it was not sufficient to log the underlying commands invoked. We wanted to account for visual complexity and therefore defined a function as a graphical element on which the user could act, rather than an underlying piece of code. Functions are therefore action possibilities (affordances) that are specified visually to the user[15].

The following heuristics were developed to count the functions in the default MSWord interface:

[13] McGrath [1994] provides a comprehensive overview of methodology in the social and behavioural sciences. He noted that all methods are valuable but have weaknesses. However, one can offset the different weaknesses of various methods by using multiple methods.

[14] There are clearly other factors contributing to users' experience of complexity, for example, the actual domain of the application can be inherently complex (e.g., engineering design is the domain of CAD software) or the structure of features within the interface can play a role (e.g., progressive disclosure). In this study we focused specifically on the number of functions.

[15] Defining functions as action possibilities is similar to what Hsi and Potts [2000] refer to as the morphological view of functions.

- Each *final* menu item in the menus from the menu toolbar counted as one function. A menu item was not considered final if it resulted in a cascading menu.

- Each item on a toolbar counted as one function (even if it provided a drop down menu – e.g., **Borders**, **Styles**, and **Font Colour**).

- Each button on a scrollbar counted as one function except the scroll widget (composed of scroll left/up and scroll right/up buttons as well as the bar itself) that in its entirety counted as one function.

- The selectable items on the status bar each counted as one function.

Using these heuristics we counted 265 functions in the default MSWord interface. The second-level count was independent of the first count and included all options available on the first-level dialog boxes[16] – 709 options were available. The heuristics used for counting dialog box functions are considerably more complex than those in the main interface and are described in Appendix B. In this chapter we report only on the first-level functions (n= 265).

3.1.2 Participants

The sample consisted of 53 participants from the general population. All participants were MSWord users. As we did not have a simple random sample of the population of MSWord users, care was taken to include subjects from a variety of occupations and organizations and from across the organizational hierarchy. Variation in terms of age, gender, and education and experience using computers generally and MSWord in particular were also attended to. An aggregate description of the participants is given in Table 3-1. A few observations can be made. We did not have an equal number of males and females – approximately three fifths of the participants were female. On average, the participants were quite highly educated in that a rating of 5 indicates the completion of an undergraduate degree. The average level of MSWord expertise was low in that a rating of 2 indicates moderate knowledge. The average number of years that the participants had been using MSWord was 4.7, however, the large

[16] First-level dialog boxes are those that are accessible directly from a menu item.

standard deviation of 3.5 indicates that some participants had been using MSWord for only a short period of time and others had been using it for many years. Although it is not shown in the table, the daily usage data included 2 participants who reported spending less than 15 minutes per day word processing, 10 less than an hour, 22 between 1 and 3 hours, 14 between 3 and 5 hours, and 6 more than 5 hours a day. Table 3-2 shows a breakdown by occupation and gender.

	Gender		Mean age (/5)	Mean education (/7)	Mean MSWord expertise (/3)	Mean # years using MSWord
	M	F				
Total	19	34	2.7 = late 30s (1.2 SD)	5.6 (1.5 SD)	1.7 (0.7 SD)	4.7 (3.5 SD)

Table 3-1: Aggregate description of participants (n=53).

	Male	Female	Total
Admin assistant	-	9	9
Journalist	2	-	2
Secretary	-	8	8
Academic	2	-	2
Professional (non-tech)	-	7	7
Technical expert	8	1	9
Student	-	1	1
Designer	-	1	1
Admin manager	4	2	6
Lawyer	2	2	4
Librarian	1	3	4
Total	19	34	53

Table 3-2: Occupations of participants (n=53).

3.1.3 Instruments and Data Collection

Functionality Interview

The objective of the functionality interview was to establish empirically (1) the distribution of users in terms of their familiarity with the functions in MSWord and (2) the use of functions and the variation in use across users.

A researcher presented each participant with two series of printed screen captures. (A sample screen capture is given in Appendix C.) The first set successively revealed all the first-level functions on the default interface. The second set was a simple random sample of all the dialog box functions. For each function, participants were asked:

1) Do you know what the function does?

And if so,

2) Do you use it?

Responses to question one were scored on a 2-point scale: *familiar* and *unfamiliar*. Responses to question two were scored on a 3-point scale: *used regularly*, *used irregularly*, and *not used*. To serve as a reminder, a sheet showing those response scales was placed in front of the participants (Appendix D). Participants were told that familiarity with a function indicated a general knowledge of the function's action but that specific detailed knowledge was not required. A regularly used function was defined as one that was used weekly or monthly and an irregularly used function was one that was used less frequently. An instrument was used to facilitate the recording of this data (Appendix E). Methodologically it is important to note that while that instrument appears to be a checklist, it was not a self-completion instrument. It was administered by the researcher and was in practice the focus of an extensive conversation about what was liked, used, hated, etc. There was thus little potential for response bias[17], a well-understood problem with self-completion instruments.

We were particularly concerned with ecological validity. The most valid way to assess the familiarity and use of functions by our participants would have been to have them use their own system while reporting familiarity and usage. However, we were concerned that if they had customized the interface in any way there was the potential to introduce error in the recording and make comparisons problematic. Our approach was to use colored screen shots of an "out-of-the-box" version of the word processor. We did, however, take screen captures from our participants' machines so that we could later assess the extent to which they had customized their interface.

To discourage guessing in order to ensure reliability, participants were told at the outset that they would be asked periodically to describe the action of any function with which they

[17] Response bias is a type of non-sampling error caused by respondents intentionally or unintentionally providing incorrect answers to research questions.

reported familiarity. Unreliable participants could in this way be identified and data discarded. Fortunately this did not turn out to be an issue in this study[18].

In-depth Interview

An in-depth interview was conducted with each participant to both ground and extend the quantitative work. Here specific issues that had been raised in the *Functionality Interview* were probed and participants were encouraged to talk more generally about their experiences with word processing overall, and MSWord, in particular.

The *Functionality Interview* and the *In-depth* Interview together required approximately 1.5 hours of each participant's time.

Questionnaire

Prior to the interview each participant was given a poll-type questionnaire, *Experiencing Word Processing* (Appendix F). This took approximately 30 minutes to complete. It included a series of questions on work practices, experience with writing and publishing, the use of computers generally, and the use of word processors specifically. A number of questions were designed to gather information for scale construction for the evaluation of Microsoft's profiling study. Basic demographic information, such as age, gender, education, and occupation was also requested. Throughout the questionnaire open-ended responses were encouraged and space was provided.

3.1.4 How Our Work May Be Differentiated from that of Others

Our work can be differentiated from earlier research in a number of ways. First, we used multiple methods and the results show this provided us with a unique perspective for thinking

[18] We made an attempt to systematically check for reliability, but with most subjects the conversation about the features was rich and detailed which made it difficult to do a systematic check. For example, features were sometimes mentioned out of order as they related to other features and participants often qualified their responses without prompting – "I love that because…", "I hate that because...", "I use that for...", and "I prefer to use the hotkey". Although having a numeric reliability measure might make the data appear more valid, there was a trade-off, as is often the case, between "getting the numbers" and "capturing the narrative". Having said all this, there were 243 occasions across all participants when they were asked to describe a function and 236 out of the 243 (97.12%) were described correctly. No participant described more than 2 functions incorrectly.

about complex software. The different approaches did not provide conflicting answers, but rather helped to elaborate an extremely complex question, each highlighting a specific dimension of the problem. Quantitative methods offered a detailed description of the feature space, questionnaire responses helped identify patterns and summarize the data, while the in-depth interviews allowed us to probe these summary statements in order to understand the users' actual experience and how it varied. Second, our data were self-reported—from questionnaires and in-depth interviews—in contrast with logging, which has generally been the method of choice in computer science. It is important to note that while logged data report the functions actually used (at least as measured in terms of keystrokes), this method cannot distinguish between *familiarity* and *use*. Finally, while the reliability of self-reported data are dependent on the participant's honesty and ability to recall information, information on a function that is used irregularly could be missed if logging is not carried out for an extensive period of time. Thus, by using a variety of methods we compensated for the limitations of each method used on its own.

3.2 Results

3.2.1 Quantitative method: Counting functions

Software could be defined as "bloated" when a significant proportion of the functions available are not used by the majority of users.

MSWord was indeed found to be "bloated" by this definition, as the pie charts illustrate (Figure 3-1). Compare the number of functions used by various percentages of the users and the functions used regularly. Of the 265 first-level functions, 15.8% (42) were not used at all and only 21.5% (57) were used by more than half of the participants. There were only 3.3% (12) functions that were used regularly by more than three quarters of the participants.

By looking at the number of functions with which users were familiar, however, we see that the distribution is much more even and that users were familiar with a great deal more than they actually used. Note that this familiarity data could not have been captured through logging. There was only one function that none of the users could identify, namely, **Extend**

Selection (EXT) on the status bar. Twenty-eight percent (74) of the functions were familiar to 13 or fewer participants (1-25%), and 26% (69) of the functions were familiar to 40 or more (76-100%) of the participants. The high degree of familiarity points to one way in which this narrow definition of "bloat" may mask the actual experience of the user. Familiarity may lead to a certain level of comfort and so if users are familiar with functions, even if they do not use them, they may be less likely to perceive these unused functions as "bloat" rather than as functionality they simply do not use.

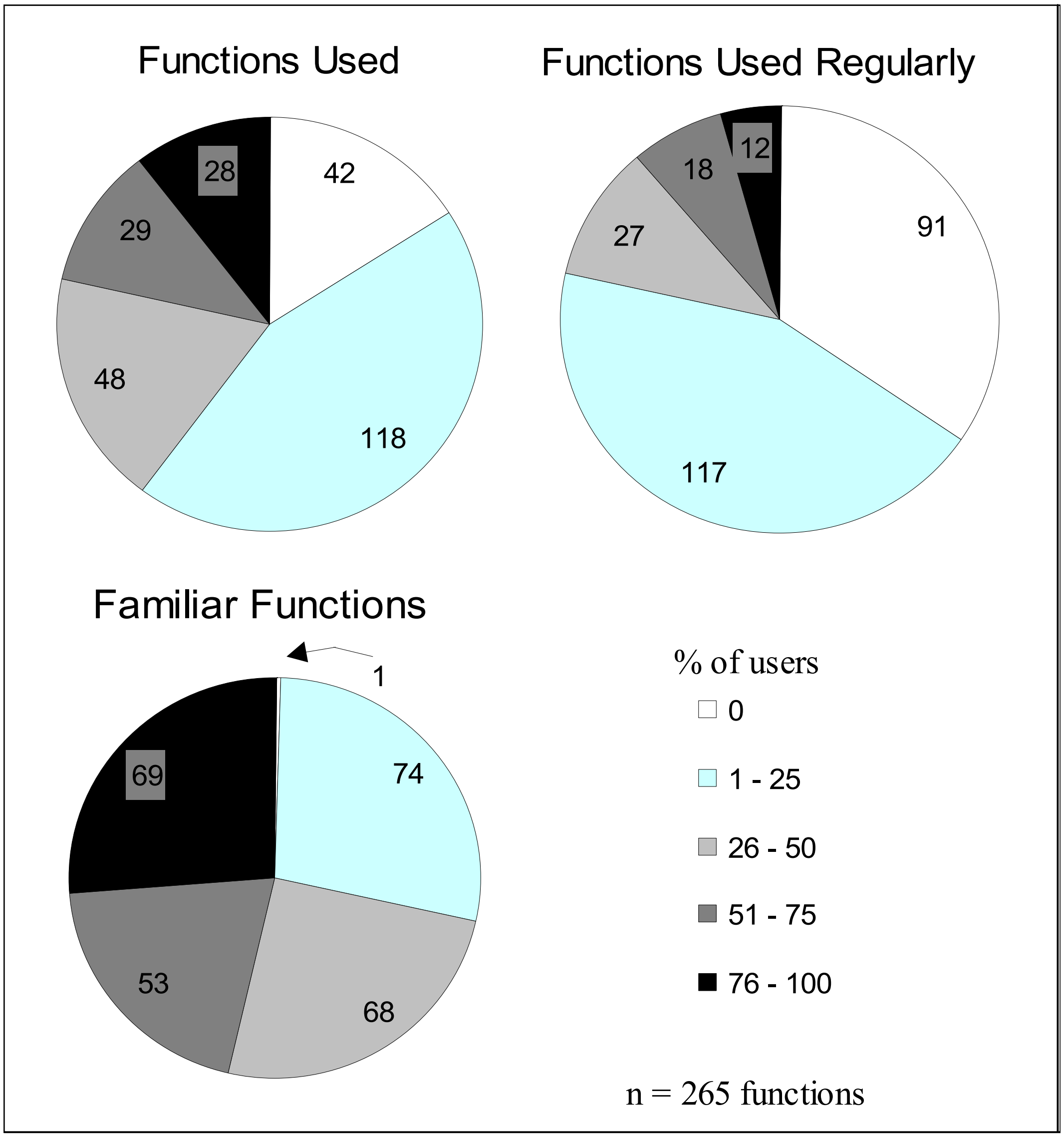


Figure 3-1: Number of functions that were used (both regularly and irregularly), were used regularly only, and were familiar to our participants (n=53 participants).

We can also look at the relationship between familiarity and use from the perspective of the individual user. Figure 3-2 shows a comparison for each participant between the percentage of functions with which they were familiar and the percentage actually used. On average, the Usage to Familiarity Ratio was 57% (SD=0.15). Figure 3-3 shows the same data sorted in descending order of use.

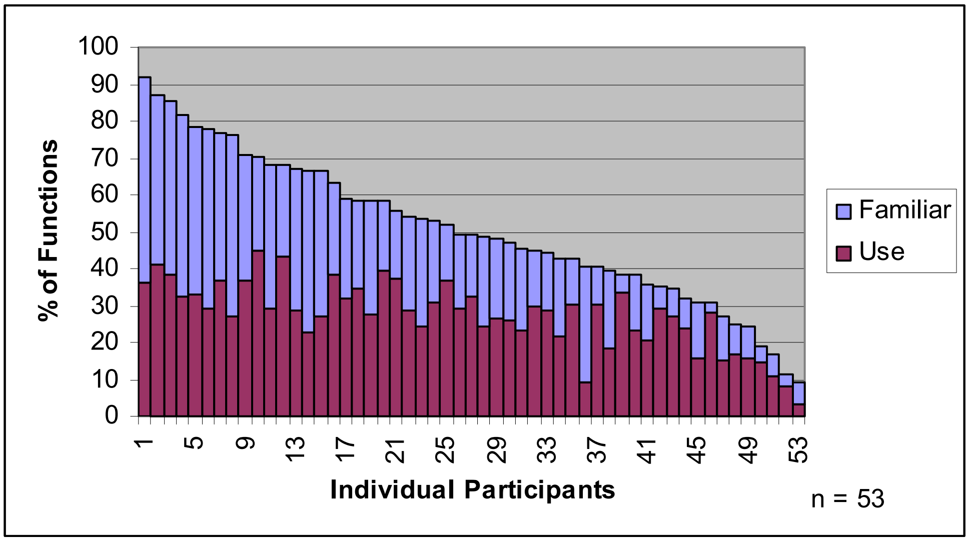


Figure 3-2: Percentage of functions "familiar" and "used" for each participant sorted in descending order of familiarity.

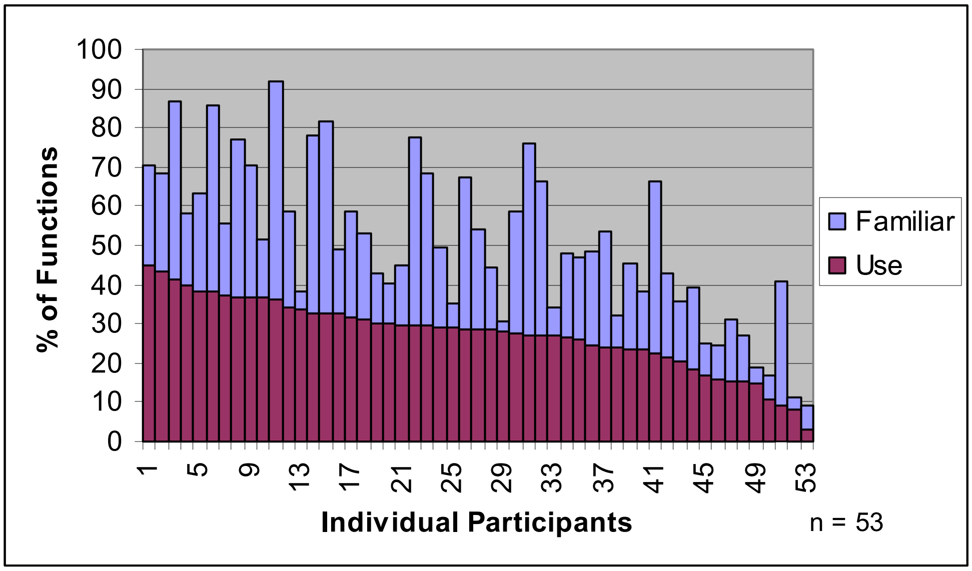


Figure 3-3: Percentage of functions "familiar" and "used" for each participant sorted in descending order of use.

Table 3-3 shows that a relatively low percentage of the functions were actually used; on average the participants were familiar with 51%, and used 27%[19]. There was greater variation in the number of functions with which participants were familiar (range from 9% to 92%) than the functions actually used (range from 3% to 45%). Figure 3-2 shows this graphically. However, this data cannot tell us whether each user experienced the unused functionality in the same way, or whether or not they experienced it as "bloat".

	Number of First-level functions		
	Minimum	Maximum	Average
Familiar to participants	24 (9%)	245 (92%)	135 (51%)
Used both regularly and irregularly by participants	8 (3%)	119 (45%)	72 (27%)
Used only regularly by participants	6 (2%)	82 (31%)	40 (15%)
Used only irregular by participants	2 (1%)	69 (26%)	32 (12%)

Table 3-3: Means and ranges of "familiar" and "used" functions (n=53).

3.2.2 Qualitative Method: The Users' Experience

Our quantitative analysis showed that there was a great deal of unused functionality, but is this "bloat", or more neutrally, simply unused functionality? While our quantitative analysis highlights many interesting aspects about the feature space, it cannot shed light on this question. Yet, the answer to this question has important implications for design. In this section we first focus on data from the questionnaire to answer the following questions:

1) To what extent did users report that they were satisfied or dissatisfied with their word processor?

2) Was concern with unused functionality a major source of dissatisfaction for users?

3) What other sources of dissatisfaction with their word processor did users report, and how can these be categorized?

[19] We found that participants' familiarity and usage at the dialog-box level was roughly half of what it was with the first-level functions: participants were familiar with on average 28% of the dialog box functions and used 13%, again with a ratio of about 2-to-1 between familiarity and use.

4) Did users actually perceive their word processing software as "bloated" and did they use this language?

5) What impact did unused functionality have on usability?

After answering these questions we will report what the users themselves said. We hear what frustrated them, how they responded to new versions of MSWord, and how in the final analysis they got their work done. It is important to note the ways in which these two methodologies complement each other, in particular how in-depth interviewing was able to deepen our understanding of users' experience and suggested design options that are masked by other methods.

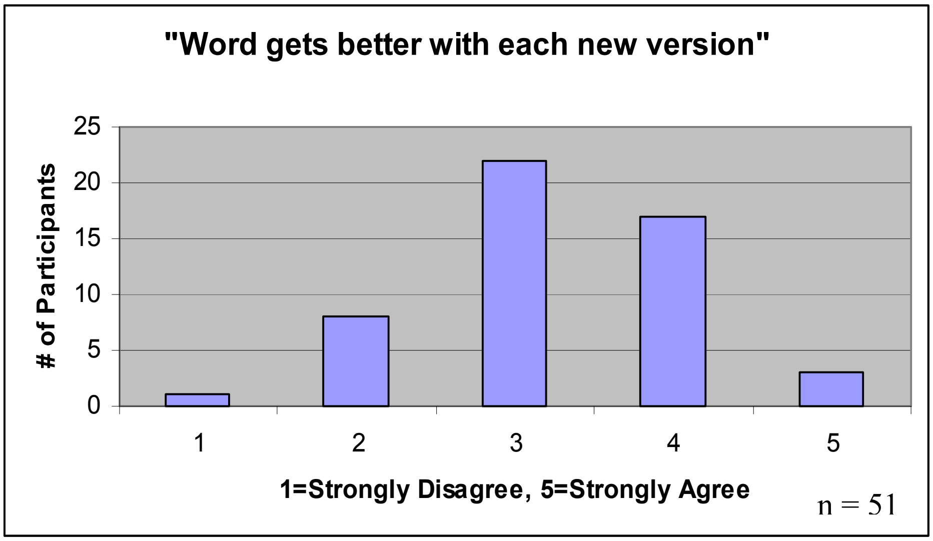


Figure 3-4: Participants satisfaction with MSWord.

Questionnaire

Participants were asked to rank on 5-point Likert scale (Strongly Disagree, Disagree, No Opinion, Agree, Strongly Agree) a series of 29 statements about their perception of using MSWord (Appendix F, Questions 33 and 34). Figure 3-4 shows the result for one of these questions, and is representative of the findings for statements on general satisfaction/dissatisfaction: the majority of the users reported "no opinion" to these statements and the rest of the users were almost evenly divided between those who agreed and those who disagreed. However, when participants were asked in the interview to discuss these statements, they were, in general, more satisfied than the questionnaire responses indicated. Furthermore, the interviews revealed that the major source of dissatisfaction was not about the large number of functions, or concern that the extra functions were getting in the way. Rather concern centered on factors such as poor implementation, unpredictability, and inconsistency. Finally, with respect to "bloat", there was not a single person in our study who used the term, either in written comments on the questionnaire or in the interviews.

The responses to questions designed to assess the impact of a large number of functions on usability did, however, suggest that a problem existed. Examples of these are in Table 3-4 below. Users were almost evenly divided between those who agreed, disagreed or had no opinion when asked if they were overwhelmed by the number of interface elements. However, when asked specifically about the impact of excess functionality on their activities they were more clearly divided.

	Agree	No Opinion	Disagree
I am overwhelmed by how much stuff there is. (n=51)	27.5%	39.2%	33.3%
I have a hard time finding the functions I need unless I use them regularly. (n=53)	58.5%	5.7%	35.8%
After using a new version for a short time, the commands and icons that I don't use don't get in my way. (n=51)	51.0%	17.6%	31.4%
Wading through unfamiliar functions can often be annoying/frustrating. (n=53)	62.3%	17.0%	20.8%

Table 3-4: Responses to statements about usability.

But how would users like to see excess functionality handled? Again, our participants were divided, and offered no easy solutions for designers (Table 3-5). Only 24.5% wanted to have unused functions removed entirely but 45% preferred to have unused functions tucked away. The fact that 51% wanted the ability to discover new functions as they use the application points to one underlying reason for users not wanting unused functions removed.

	Agree	No Opinion	Disagree
I want only the functions I use. (n=53)	24.5%	9.4%	66.0%
I prefer to have unused functions tucked away. (n=53)	45.3%	15.1%	39.6%
It is important to me that I continually discover new functions. (n=53)	50.9%	18.9%	30.2%

Table 3-5: Users' preference for number of functions on the interface.

But why were some users not bothered by excess functionality while others were? This is, we argue, in part a reflection of the diversity among the users of word processors in the general population. Furthermore, the power of today's word processor is that it is not necessary to be a technical expert to use it, however, lack of computing expertise may limit a user's ability to critique it.

A fascinating finding was that a specific occupational group, the secretaries and administrative assistants, reported the greatest satisfaction with their word processor. Initially we thought this might be explained by the fact that they would have received more training than members of other groups and were the heaviest users, but the results of the questionnaire did not support this. However, in the analysis of the transcripts from the in-depth interviews we did notice a pattern. When participants from this group were asked to say more about what they specifically liked about MSWord, or how it could be improved, they had difficulty giving precise answers. A number said that they simply accepted it as it was, or assumed that this was "just the way word processors are"! So, while this group included heavy users of word processors, they appeared not to have sufficient computing experience to think critically about this tool. When we looked at the technical experts, we found that not only did they report the greatest dissatisfaction, they were able to articulate this in terms of specific problems and underlying issues. The computer scientists, in particular, had a good understanding of what is possible in a software application and were less accepting of "bad" design or what they saw as "sloppy implementation".

These observations raise two points. First, designers should not automatically assume that users can answer their questions accurately (even if the questions are well designed). Second, there is a limitation to survey methodology if the goal is to understand user experience. We now turn to the users' own accounts given in the in-depth interviews. These more nuanced accounts of user experience suggest some new design ideas.

In-Depth Interview

Source of Dissatisfaction: Excess Functionality

In this section the users speak. It is important to understand that the analysis that runs through this section comes out of a detailed reading and summarizing of the transcripts from the in-depth interviews. The individual quotations have been selected both because they are representative of the analytical point we are making and for the way in which they give life to the issues under discussion.

A number of participants stated explicitly that their needs could be met by a "simpler" system. A senior technical expert commented:

> *I want something much simpler… I'd like to be able to customize it to the point that I can eliminate a significant number of things. And I find that very difficult to do. Like I'd like to throw away the 99% of the things I don't use that appear in these toolbars. And I find that you just can't, there's a minimum set of toolbars that you're just stuck with. And I think that's a bad thing, I really believe that you can't simplify Word enough to do it.*

Another senior technical expert said it the following way:

> *I've heard a lot of people around here ... saying that they actually are looking for something really basic, like a streamline, like the Ford Escort version as opposed to the Cadillac version of the word processor… it's like quickly type/bang out a simple document and that's it, right? ... I think there's a bit of a VCR complex or too many buttons and whistles. I just want to have the on/off, rewind, forward buttons.*

By way of contrast, a junior technical expert suggested that he did not want a simpler system, but that he was concerned with the amount of screen real estate that these functions took up:

> *I don't think they should be eliminated. It's always good to have them, but they shouldn't be given the same prominence or real estate on the screen as the other options.*

An older office administrator who had once studied mathematics suggested that there was a need for a "light" version of the software not because of limited screen real estate but to prevent confusion:

> *I think maybe what they could do is have different levels so you would not be bogged down with so many features, and if you don't need them all, they are just really in the way and they get cumbersome. They get into something you don't really want. So if they had something like Basic Microsoft Word, ... that would be useful ... for people who just do letters or something like that. I think that there is, in a way, too much there and that for a user like me*

> *it is, in fact, a disadvantage – you get lost in it and when you need something very quickly, which is usual, whether one is typing a letter or a document, you don't have oodles of time to go exploring for 3 days. And then it takes you half an hour to do the thing… My background was mathematics, where you learn basic stuff and [then] you learn more difficult stuff and you build it. It goes very logically, whereas here you are just thrown in the middle of it and you flounder.*

Several participants specifically said that they wanted all the functions present even if they did not use them because they might want them some day. This points to an underlying apprehension that functions that are not visible might get "lost" if they were tucked away. A young female lawyer who relies heavily on her secretary said:

> *No, I think I prefer to have it there just in case there's an occasion when I'm here, she's [her secretary] not here and I want to do something. Then I'll just go in and do it. Like, an example would be page numbering, or making a list, or using the automatic numbering, or putting bullets in, or some sort of formatting thing that I want to do if she hasn't done it and we've got to get the document out. So, no, I think it's fine how it is, and in fact, I sort of like to have the option. I wouldn't like to be treated like I only can work at a certain level. I prefer to have the option to work at a higher level if I choose to.*

Others wanted all the features present specifically because they saw it as a sign of being up-to-date. A young female consultant said:

> *And I always want the latest version whether I [laughs] you know, really use all the new stuff… I want it, whether I know what to do with it or not. [laughs] I understand why some people would only want to use what they have, 'cause that's what they're familiar with, but now you just can't be like that anymore.*

And others pointed out that reducing the functions would impede their ability to learn through exploration. A graduate student in the humanities said:

> *Yes, it would probably be useful … hmmm, I have two answers to that … To some degree it would be useful to have a reduced set, but I like sometimes just playing around and discovering a function. So if you only have a reduced set then you don't have much chance to*

> *accidentally find a function and say "ah, this is what this does" but the results [of the functionality interview] show that I haven't been experimenting that much!*

Not all the participants were concerned about the large number of functions. In fact, several were adamant that they liked to have them all. An entrepreneur and owner of a small successful IT company put it this way:

> *And I know that in some software packages they try to create simple menus to allow you to decide what you want to see. I always default to the full set and the reason is that I feel comfortable with technology and I'd rather see the full extent of what's available, not just the sliver that is usable by me. I don't want to see a sliver or portion of it that has been determined statistically to be more useful for most users and think that I'm missing out on some features.*

The general sentiment expressed in the interviews with respect to the number of functions available can be summarized into the following three observations:

Observation 1: Many participants expressed frustration with having so many unused functions. The dominant reasons for frustration were the desire for something simpler and to reclaim screen real estate. To counter this, some participants seemed perfectly content to have a vast selection of functions.

Observation 2: Although some participants would be content with a "light" version of MSWord, the dominant feeling was not to have unused functions removed from the application entirely. The main reasons against a light version were the apprehension of a total loss of unused functions, and the perception of only being able to work at a certain level.

Observation 3: Some participants used exploration of the interface as a means of learning the software. They felt that if unused functions were eliminated entirely, this would limit their ability to learn through exploration.

Clearly there are conflicting views. The challenge to designers is to accommodate the diversity of user needs. In these quotations we begin to see what motivates users and how some of these motivations tie into underlying values, e.g., speed of accomplishing tasks, and the need to be seen as up-to-date. We also see how exploratory learning is one consequence of keeping all the functions at hand.

Additional Sources of Dissatisfaction

A source of dissatisfaction for some users was their perception that the multiple ways of executing an operation were confusing. In some cases this was because these variations did not produce the same result (e.g., print menu versus print icon) and users perceived this as a sign of inconsistency. Others perceived the fact that variations produced the same result (save menu versus save icon) as a sign of unnecessary redundancy. A government policy analyst noted:

> *It seems very redundant to have that many different ways of doing something, and it makes training very confusing when you're just starting up with the software.*

As we have noted there are several sources of dissatisfaction for users, and that excess functionality is seen as problematic by some. While different users "liked" or "hated" different features, there was almost universal distain for automatic features. For a variety of reasons, participants from across the occupational categories complained about these. As one technical expert exclaimed: "*Don't go there*!" or a librarian, commenting on AutoCorrect[20] put it this way:

> *What I type is what I want, so I don't want the machine second guessing me.*

For others the problem was compounded because they did not know how to turn these automatic features off. A female consultant complained:

> *With each new version there's a tendency for it to try and predict what you will do next. Drives me nuts! It's an advance that is totally annoying.*

[20] This function corrects a word automatically while it is being typed.

The angst expressed about the automatic features was clear evidence of the fact that users need to have ultimate control over their interface.

But, even here there were a few who perceived that the automatic features that they used worked well and they considered them timesavers.

3.2.3 Summary

The analysis of the questionnaire data, informed by the in-depth interviews allowed us to further specify "bloat" by identifying an *objective* and a *subjective* dimension. There is a subset of features that are not used or wanted *by any* user which we designate as *objective bloat*. The remaining features are divided into two sets which vary from user to user and are thus subjectively defined. One set includes those features that are not used *and* not wanted by an individual user — this is *subjective bloat* for that user. The second set includes those features that are wanted by that individual user, whether or not they are actually used. To refer to the unused functions in the second set as "bloat" is misleading as the user's experience of this unused functionality is not negative. In fact, users' responses both to the presence of unused functionality, and how they would like it handled varied widely. This discovery of a set of unused features, both wanted and unwanted, that is subjectively defined by each user, opens the design space and raises new challenges for interface designers. There is certainly more to "bloat" than meets the eye. Some implications for design are discussed later in Section 3.4.

3.3 Evaluating and Extending Microsoft's Study on User Profiling

The third method we used was user profiling which, like our second method, is qualitative in nature. We earlier introduced the Microsoft profiling approach in Section 2.3. Recall that based on Microsoft's informal focus group of 12 intermediate and expert users it was found that users could be categorized into one of two profiles, A or B, depending on their perception of software as "bloated" or not. Profile A users prefer software that is complete, they will stay up-to-date with upgrades, they assume that all interface elements have some value, and they blame themselves when something goes wrong or when they can't figure out how to perform a specific task. Alternately, Profile B users prefer to pay for and use only what they need, they are suspicious of upgrades, they want only the interface elements that are used, and they blame the software and help system when they can't do a task. A user's profile was independent of expertise.

The Microsoft profiling approach, although unpublished, represented a reasonable first attempt at understanding the perceptual component of "bloat". Our goal was to attempt to reproduce these findings in a more systematic study with a sample of users from the general population.

Our results showed partial support for the existence of the A and B profiles. We had to discard the questions on blame as few in our sample were willing to blame either the software or the company when they had problems, and even fewer were willing to blame themselves, only 3.8%! On all the questions relating to blame we had a high reporting of "no opinion", and at least 2 participants said that "blame" was very strong language and that they felt discomfort with the term. We have no explanation for why this was not a problem in the Microsoft group, other than a possible cultural one.

After removing blame we had three scales to construct. They are summarized in Table 3-6 along with the Cronbach's alpha[21] (α), a reliability measure.

Functions	"The number of functions in the in interface does not make it difficult for me to find the function I am looking for." (# Variables = 3, α = .83)
Up-to-dateness	"I want my MSWord software to be up to date – I want the latest version." (# Variables = 3, α = .77)
Completeness	"I want a complete version of MSWord even if I don't use all the functions." (# Variables = 6, α = .76)

Table 3-6: Summary statement for three scale variables (n = 50).

[21] Cronbach's alpha measures how well a set of items (or variables) measures a single uni-dimensional construct. When data have a multi-dimensional structure, Cronbach's alpha will usually be low [SPSS Frequently Asked Questions: What does Cronbach's Alpha Mean? 2001]. Cronbach's alpha can be written as a function of the number of test items, N, and the average inter-correlation among the items, r, such that $\alpha = (N \times r) / (1 + (N-1) \times r)$. Nunnaly has indicated 0.7 to be an acceptable reliability coefficient but lower thresholds are sometimes used in the literature [Nunnaly, 1978, cited in Santos, 1999].

The 10 statements (variables) that make up these three scales were are also found to "hang-together" (Cronbach's α of 0.81) which allowed us to aggregate these scales to comprise what we call the Feature Profile Scale. See Appendix G for a detailed description of how the Feature Profile Scale was constructed.

The distribution of the cases across the Feature Profile Scale revealed that while there were concentrations at both ends of the scale there was a substantial group near the center. We therefore divided the subjects into three equal-sized groups, which we distinguished as the *feature-keen* (Microsoft's Profile A users), the *feature-neutral*, and the *feature-shy* (Microsoft's Profile B users). The distribution is shown in Figure 3-5.

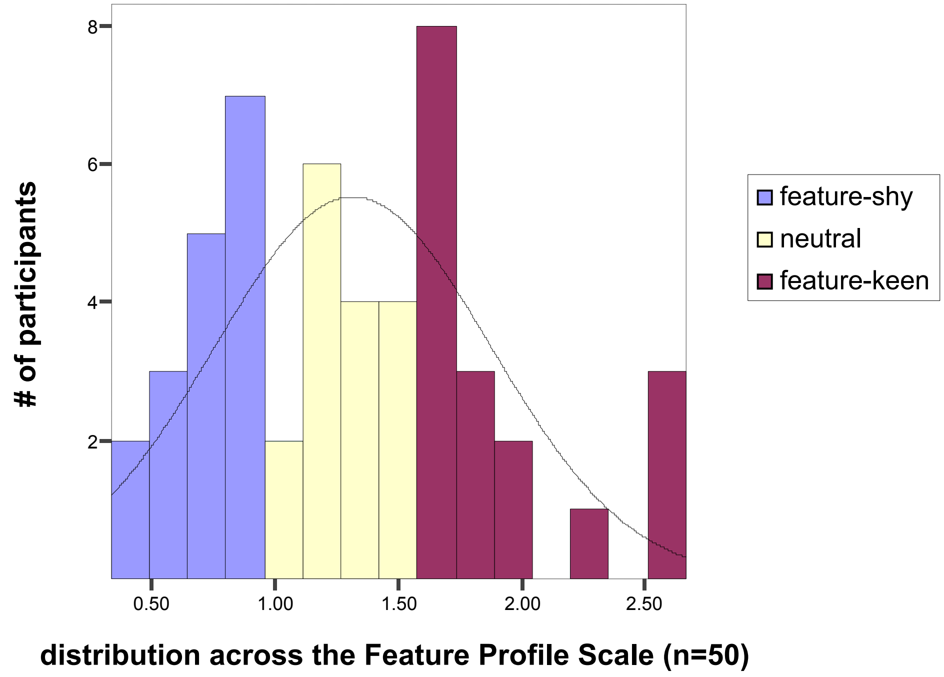


Figure 3-5: Distribution across the Feature Profile Scale. The ranges for each of the three groups are: feature-shy [0. 0.92), feature-neutral [0.92, 1.58), feature-keen [1.58, 3.00].

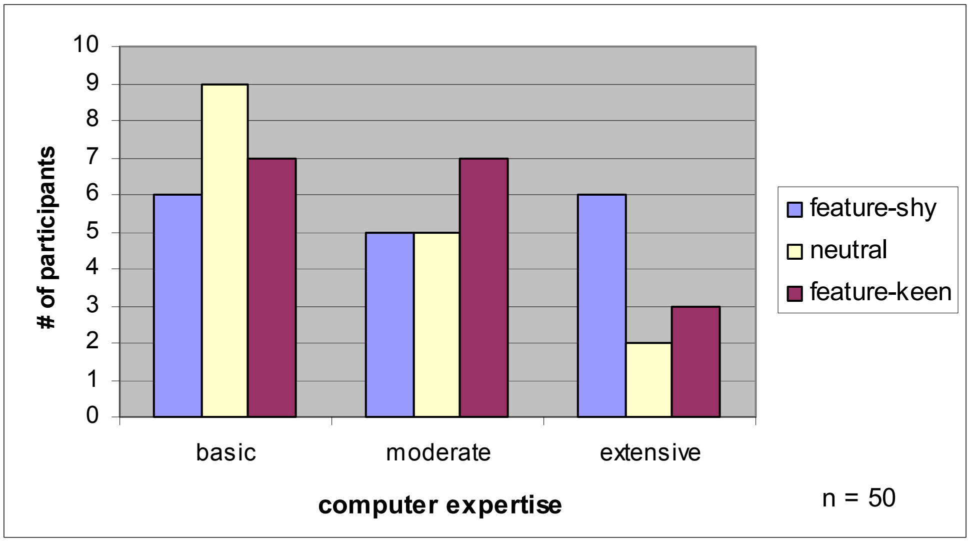


Figure 3-6: Distribution of computer expertise the Feature Profile Scale.

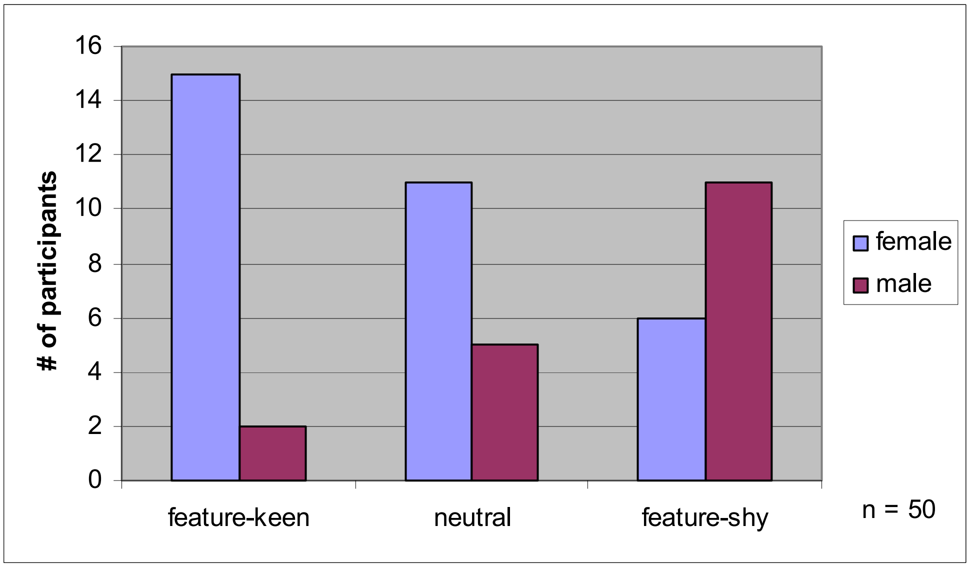


Figure 3-7: Distribution of gender over the Feature Profile Scale.

We found support for Microsoft's finding that the perception of "bloat" is independent of expertise (Figure 3-6). In addition, we found that it is independent of both the number of functions used and the number of familiar functions.

While there was some support for the profiles; that is, the perception of "bloat" varied between groups of users, we wondered what else this profile might be capturing. First, we thought it might distinguish early from late adopters and that gender might play a role in explaining the outcome. Figure 3-7 shows that there was indeed a difference based on gender, but contrary to our expectations it was the females who were the feature-keen, those wanting the most up-to-date and complete version of the software. However, when we looked at the distribution of the occupation of our participants (Figure 3-8), an alternate explanation was suggested. The gender difference also reflected differences in terms of position in the division of labor.

The females clustered in administrative assistant and secretarial occupations, those for whom word processing may be the base of their craft knowledge and a source of status. The feature-shy included the journalists and lawyers, all of whom have staff to format and/or edit their work. As they themselves are not responsible for the look and feel of the final product, they may not feel a need to upgrade. Also included in the feature-shy were the technical experts and one of the two academics, both groups who are less likely to have support staff and for whom word processing is yet one of many tasks, and a secondary one at that. They are the most overburdened, responsible for both the creation and production of documents. Not surprisingly, perhaps, they were the least interested in new versions of the software. This finding is consistent with Mackay's [1991] finding that technical experts are not interested in exploring powerful new features and would rather focus on task completion.

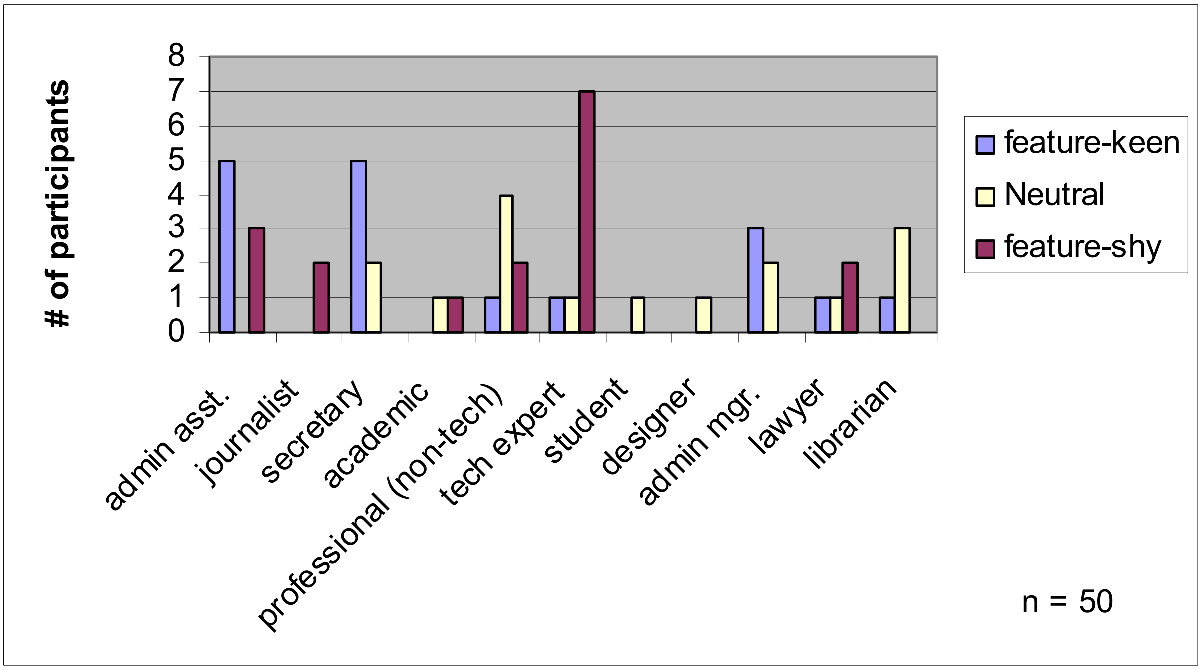


Figure 3-8: Distribution of occupation over the Feature Profile Scale.

3.4 Bringing it All Together: Directions for Design

We had become concerned that the recent use of the term “bloat” in the popular press and more importantly in the technical community could suggest that widely used general applications, such as word processors, are filled with unnecessary features. We were especially concerned that the potential for these reports of user dissatisfaction could lead to the conclusion that users would be better served by simple or light versions of these applications. What was missing were the research studies that tested and evaluated these assumptions. It was a goal of our study to begin to fill this gap, and to be able to understand more fully how users actually *experience* a complex application such as a word processor.

Our first objective was to see if the labeling of this complex software as “bloated” had any validity. If unused functionality was the metric, MSWord was clearly “bloated”—a little over 50% of the functionality with which users were familiar was actually used. And, while users were dissatisfied with a number of aspects of MSWord, not all this dissatisfaction centered on excess functionality.

“Bloat” can be further specified and defined in terms of *objective* and *subjective* “bloat”. In terms of *“objective* bloat”—functions used by few users—we have the following design recommendations:

- Eliminate unused functions.
- Relocate functions used by only a few users from high-level visibility in the interface. The determination of the exact “cut-off point” is likely application-specific.
- Prevent objective bloat. This requires a shift in design practice from programmer/marketing-centric to human-centered design. There is a need to recognize that there is a cost to the user of unused features. Each function should be evaluated carefully before it is added to the interface.

This leaves a subjectively defined group of unused functions. But this subset cannot de facto be defined negatively – this is not what all users told us. Subjective “bloat” is thus defined as the particular subset of functions that are not used and not wanted by an individual user. The fact that not all of an individual user’s subset of unused functionality can be labeled as “bloat” was an unexpected finding of our study. Subjective “bloat” varies from user to user, so creating a simple or basic version by eliminating functions will inconvenience most users, albeit in different ways. To put it another way, my favourite function may be your “bloat” and vice versa. We hope that this redefinition will encourage a more nuanced understanding of the richness of heavily featured software applications and that a catch-all phrase such as “bloat”, which distorts users’ experiences, will no longer be used.

What is exciting is how this understanding opens the design space, challenging designers to accommodate both functionality that is used and functionality that is unused but nonetheless still wanted. The ultimate goal might be for each user to have an interface that includes functionality suited to his/her needs and desires, yet does not limit access to additional functionality. Although this goal is not likely achievable in its purest form, some progress can still be made.

Interface design has begun to acknowledge that “one-size-fits-all” interfaces may not in fact fit all. Facilities for customization and tailoring are included in most complex software applications, however, the high overhead required to customize renders them neither effective nor adequate. We argue that the philosophy of design needs to move away from “enabling the customization of a one-size-fits-all interface” to supporting the creation of a personalizable interface. The personalization solution needs to be lightweight and low in overhead for the user, yet not limit or restrict their activities. We suggest that multiple interfaces may be one way to accommodate both the complexity of users’ experience and their potentially changing needs. Individual interfaces within this set would be designed to mask complexity and ideally to support learning. We recognize that continual access to the underlying formatted document or text needs to be preserved.

The concept of a multiple-interfaces design that includes one or more personalized interfaces is quite general and could be applied in many different ways. The basis for differentiating the interfaces and the number of interfaces included can be viewed as parameters to an extensive design space. In this dissertation we use reduced functionality sets as the basis to differentiate the interfaces. One could also have task-based interfaces, and although further definition would be required, one could consider interfaces defined by psychological stereotypes, social roles, or digital personas.

3.5 Towards Study Two

The findings from Study One suggest that some form of personalization is a reasonable research direction for design. As discussed in Section 2.4, there are two fundamental ways to achieve a personalized interface. The first is to have the user customize or adapt the interface him/herself in such a way that meets his/her needs. The second is to have the system change the interface dynamically based on what it knows about the user. These approaches can be juxtaposed as adaptable interfaces versus adaptive interfaces. Both approaches have design challenges: customization facilities have traditionally been too complex for users and therefore users tend to adapt their interface very little; adaptive solutions often result in a loss of user control and therefore user dissatisfaction. Thus, an effective method for personalizing technology remains elusive. Moreover, most personalization research has implicitly assumed that a personalized interface does have value, so even the value of customization is still not well established.

This prompted us step back and ask: Can we first evaluate the merits of a personalized interface independent from a method for achieving the personalization? Can we decouple the value of the interface from the personalizing mechanism? To be explicit, there are two separate but related questions to investigate:

Question One: If one assumes an ideal personalized interface – one that includes only those functions the user would like with zero overhead to the user – does this interface have value? If it does have value, do users prefer it to the default interface, can users complete tasks faster, do they feel less overwhelmed, are users generally more satisfied, do they learn the feature set more quickly, etc.?

Question Two: Assuming a personalized interface is valuable, how can we design the personalizing mechanism such that it provides sufficient control to the user but only requires minimal effort?

These two research questions will be addressed separately in the context of the Pilot Study and Study Two, respectively, in the next two chapters.

CHAPTER 4 Pilot Study – Personalization Using Wizard of Oz Methodology

The findings from our Study One suggested that a multiple-interfaces design that includes a personalized interface might be a reasonable design solution to heavily featured productivity software. In this chapter we describe the design and implementation of our first multiple-interfaces prototype and an informal evaluation conducted with 4 participants. Some of the content in this chapter has been published in the proceedings of the ACM's first Conference on Universal Usability [Baecker, Booth, Jovicic, McGrenere, & Moore, 2000].

4.1 The Prototype

For the purposes of research continuity our prototype was implemented as a front-end to the Microsoft Word, Office 2000 application.

4.1.1 Conceptual Design

The prototype included three interfaces between which the user could easily toggle. A user could work on the same document using any of the three interfaces, toggling back and forth at any time with the single press of a button.

Default Interface (DI): This contained the full functionality as offered in an "out-of-the-box" version of MSWord 2000.

Minimal Interface (MI): This contained a small subset of the functionality available in the DI, namely, the 10% of the functions from the DI that are most frequently used as reported in Study One[22].

[22] This includes: **New**, **Open**, **Save As**, **Page Setup**, and **Print** from the **File** menu; **Header and Footer** from the **View** menu; **Page Numbers** from the **Insert** menu; **Font** from the **Format** menu; **Save** and **Print** from the **Standard Toolbar**; and **Font**, **Font Size**, **Bold**, **Italic**, **Underline**, **Left Align**, **Center Align**, **Right Align**, **Justify** and **Bullets** from the **Formatting Toolbar**.

Personal Interface (PI): This contained just those functions that the user wanted.

Note that the MI and the DI remained static across all users; it was only the PI that changed for individual users.

The conceptual design did not specify a personalizing mechanism that would enable the user to modify his/her own PI. In the evaluation of this design, the PI did "grow" according to the functions each user wanted added/deleted to/from their PI, but it was the researcher who made the modifications according to a user's requests, starting with the list of functions that the users initially requested. This was the Wizard of Oz[23] component of the prototype and evaluation.

Please see Appendix H for screen captures of the working prototype.

The justification for having multiple interfaces was derived from our findings in Study One. That study showed that many users would like to have unused functions tucked away but were uncomfortable with the complete removal of unused functions. The goal of the three-interfaces design was to allow users to work with a reduced set of features knowing that access to the full system was a single click away. The ease with which the full interface could be accessed would allow for just-in-time learning. In HCI we have come to accept that providing multiple data views where appropriate is good design [e.g., Fisher & Dill, 1999; Koutsofios, 1996]. Sometimes these views can even have unique functionality associated with them. Providing these multiple interfaces was a new idea worth exploring.

[23] The Wizard of Oz technique is a common experimental evaluation mechanism used in HCI. Rather than implementing the full logic of an application, a prototype is implemented and the logic is supplied "on the fly" by a human. Often the subject is unaware that the application is not fully functional, however, this was not the case with our prototype and evaluation.

4.1.2 Implementation Details and Design

Our decision to evaluate the prototype in a field setting had a direct impact on the implementation design. The technical design criteria and rationale for the prototype were as follows:

Criterion 1: Accommodate previous customization

The prototype could not interfere with any customization that participants may have already made to their MSWord interface.

Rationale: Users might not have been willing to participate in a study if they had already made customizations to their interface and we were not able to accommodate their customizations.

Criterion 2: Easy to uninstall

The prototype had to be easy to uninstall in the event of failure.

Rationale: We expected our participants to use this prototype in the context of real work (rather than prescribed laboratory-style tasks), and therefore we had to ensure that their productivity would not be hindered.

Criterion 3: Stateless

It was desirable to make the prototype stateless.

Rationale: We wanted to avoid the situation of unexpected termination of the prototype resulting in the loss of a participant's PI settings.

Criterion 4: MI initially visible

The prototype would launch with the MI visible.

Rationale: We expected users' PIs to contain more functions than the MI, and we wanted the prototype to start with the fewest functions visible so we could tell when users desired more functionality.

The prototype was written in Visual Basic for Applications (VBA), which is the macro programming language for Microsoft Office applications. The implementation relied on the regular macro APIs available in Microsoft Word 2000. Using the Office 2000 version of MSWord was a change from Study One in which the Office 97 suite was used. In terms of the prototype itself, it could have been constructed with MSWord 97. It was the logging software, MSTracker, that required Office 2000. (See Section 6.2.2 for technical details about the MSTracker software and a discussion about why it was chosen for this study.)

The code for the prototype resided in a global template that was created specifically for the prototype. By placing the template in the MSWord startup directory of the user's file system, it would automatically be loaded whenever MSWord was launched. To uninstall the prototype one needed only to remove the template from the MSWord startup directory (i.e., move or delete the template file) and then re-launch MSWord, thus satisfying Criterion 2.

The template code added the toggle mechanism to the regular MSWord interface and added the infrastructure to support the appearance of more than one interface. The various interfaces only differed in terms of the menu and toolbar contents because these are the objects that are programmable in the VBA environment. In Study One we counted interface features such as the scroll bars, status bar, and title bar as well as the various objects that appear on these bars as functions. Because these are not programmable with VBA, they remained the same across all interfaces in our prototype and could not be personalized.

The prototype seemed to have multiple copies of the menus and toolbars (separate copies for each of the MI, PI, and DI), but in reality there was only one copy of the menus and toolbars. In order to give the appearance of there being different copies, the visibility of the items in the menu and toolbars was set to be visible or invisible according to the desired contents of the MI and PI. For the DI, all items were set to visible. This is relatively straightforward to do in VBA – the toolbar buttons and the menu items are objects which have the Boolean property **Visible**. Thus, each time the user toggled to a new interface, the code would perform a straightforward traversal of the menus and toolbars and set the **Visible** property of the items accordingly.

The information about which items to make visible/invisible was not stored in the template but rather in a separate flat file that represented the menu and toolbar structure of MSWord. The flat file was initially constructed from a participant's instance of MSWord and contained an entry for each menu item and toolbar. Thus if the user had already customized their menu or toolbar in some way, this was captured in the user's flat file. For example, if the user had added two buttons to the **Standard Toolbar** prior to participating in this study, these two buttons would have entries in that user's flat file. Thus, Criterion 1 was satisfied. Each entry in the flat file included a field that represented the visibility of the item. By setting the value of this field to a 1 or a 0 for a given item, the visibility of that item was set to be visible or invisible, respectively.

The contents of each interface available to the user, with the exception of the DI, was stored in a separate flat file with the extension of ".int". The name of the interface as it was shown in the dropdown toggle list was actually the name of the file itself. We informally called these interface files. In the case of the study conducted, each user had two interface files stored in a designated place in their file system, for example, "1 Minimal Interface.int" and "2 Joanna's Interface.int". The contents of the DI did not need to be stored in an interface file because switching to the DI simply meant that all items were set to visible and therefore there was no need to store these settings. Note that any two interface files for a given user would be identical with the exception of the visibility values and the actual filenames.

Upon startup, the template code first looked for the existence of any interface files in a designated directory. If one or more such files were found, a multi-selection toggle was added to the MSWord interface that contained the interface name (filename) for each interface file found. If more than one interface file was found, the interface names were listed in lexicographical order in the toggle selection list[24]. The first interface in the list was the interface that was displayed on startup. If no interface file was found, no toggle was added,

[24] The initial implementation used in the early stages of the Pilot Study did not list the interfaces lexicographically, but rather hard coded the MI to be listed first. During the study, however, there was a need for greater flexibility and the prototype was changed to order the interfaces lexicographically as it is described in this section.

and thus the prototype was essentially not installed even though the template code was resident.

The prototype had no notion of a personal interface per se. Rather, it recognized one or more interfaces each stored in a separate file in a designated directory. It was our labeling of the interfaces – one with "Minimal" and one with the user's name – that gave specific meaning to the interfaces provided in our study. The prototype was programmed to accept a maximum of 10 arbitrarily named interfaces.

Each time the user toggled to a new interface, the associated interface file was opened, the visibility of the menu and toolbar items were set according to the file contents, and then the file was closed. Although the prototype maintained a list of the available interfaces, it did not need to maintain the settings of each of these interfaces, which satisfied Criterion 3.

Criterion 4 was satisfied simply by setting the MI to be the interface to appear first in the toggle selection list. This was easily ensured by including the number "1" at the beginning of the filename, as described previously.

See Appendix Q for a more detailed explanation of the prototype implementation including a description of the MSWord template facility and excerpts of the VBA code.

4.1.3 Implementation and Design Challenges

There were a number of challenges faced during the design and implementation of the prototype for the Pilot Study. We document three of these in this section.

Our first design approach was to use separate copies of the menus and toolbars for each available interface. VBA does allow for the creation of new toolbars and menubars, and items can be copied from one toolbar/menubar to another. All menubars/toolbars for all interfaces could have been constructed on startup but only the copies associated with the active interface would have been visible on startup. When the user toggled between interfaces it would then only be necessary to set the entire menu and toolbars of the active interface to visible rather than the visibility of each of the items contained in the menubar/toolbars. This design might have caused a small delay on startup while all of the menubars/toolbars were created, however, it was expected that the run-time execution when toggling might have been faster. This design was not possible because it was discovered that some interface items, namely automation objects[25], cannot be copied/created in the VBA programming environment. So we had to update the visibility of each menu/toolbar item individually on each occasion that the user toggled between interfaces, as described earlier in this chapter.

We had to disable the native customization facility in MSWord (found under **Customize** option in the **Tools** menu). If users had modified their menubar/toolbar structure through the native facility, by adding or deleting an item for example, then the structure would no longer match that which is contained in their interface files. This mismatch would prevent the prototype from setting the visibility of items correctly if the user performed customization.

We encountered a few bizarre glitches in the VBA programming environment. Documenting the specifics of these faults is beyond the scope of this dissertation, however, we do provide one example. Traversing the menu hierarchy and setting the visibility of one particular menu item caused the system to crash under the following specific conditions: a user launched MSWord; before doing anything else, the user created a new document by selecting **New Blank Document** from the **Standard Toolbar**, and then toggled to the DI. The menu item that caused MSWord to hang had no discernable connection to the creation of a new document and under any conditions other than those mentioned above, it would operate normally. There appeared to be nothing specific in our code that was causing these problems but rather it appeared to be related to a problem in the VBA environment; we speculate that it was likely a memory management problem. Our only defense was to anticipate these problems and provide a work around.

[25] From the MSWord developer documentation it is not at all clear what automation objects are. There is mention of them being related to OLE objects, which conceptually seems to make sense for some of the objects like **Insert Excel Spreadsheet**. But there are many others like **Undo**, **Redo**, and **Insert Columns** which do not seem to relate to linking and embedding objects from outside the MSWord application. We speculate that any objects that don't fit in the regular MSWord object hierarchy are deemed to be automation objects.

4.2 Study Design

Conducting a formal evaluation (such as our Study Two, described next in Chapter 5) requires a significant amount of time and other resources; this includes the design and implementation of multiple data collection instruments and the construction of an appropriate sample of participants. Given that the multiple-interfaces design was new, as was the prototype implementation, we felt it was prudent to begin informal evaluation as soon as possible. The general goal of iterative design is to evaluate a design multiple times making any necessary design changes at each iteration. The expectation is that design flaws (and implementation problems) will be caught early in the design process before too much effort has been invested. The first evaluation of our prototype was thus an informal one and hence we have called it a pilot study.

4.2.1 Objectives

Our objectives for the Pilot Study were basic and straightforward:

- To explore user response to the prototype interface system. Would users like having a personal interface? What would users think about the idea of having multiple interfaces? How would they use the three interfaces? How many functions would they add to their personal interfaces?

- To collect real command usage data from an extended period of usage and build the tools to process the data, namely, to process the extremely detailed logs generated by MSTracker.

- To test the stability of the prototype and the software logger, MSTracker.

- To learn what was going to be easy/difficult, from a methodological point of view, about evaluating a prototype such as ours in a field setting.

4.2.2 Participants

There were 4 participants in the Pilot Study. Brief profiles of each participant are found below. Note that the first 2 participants were unbiased in that they were unaware of the research objectives, although they were acquaintances of the researcher (Joanna McGrenere) and others on the research team. The remaining 2 participants were Dr. Gale Moore and Joanna McGrenere who were not impartial, as they were part of the research team. An obvious apparent conflict is that Joanna McGrenere performed both the role of the researcher and a user in this pilot study. In any formal study, acting in such a dual-role would be problematic. In our pilot study, however, the objectives were very basic and the usage data was based on real tasks done over an extended period of time which would have taken considerable effort to manufacture. Having two extra participants even though they were aware of the design rationale behind the multiple-interfaces prototype was seen to add value to the informal evaluation. All participants were female. Because this was only a pilot study, we did not consider this to be a problem. For a formal evaluation, however, we would want both genders to be represented in the sample.

User 1: female, middle-aged, administrative assistant, had used MSWord regularly but used it much less during the study, proficient with general computer applications

User 2: female, middle-aged, administrative assistant, used MSWord regularly, proficient with general computer applications

User 3: female, middle-aged, executive and researcher, used MSWord regularly, highly proficient with general computer applications (Dr. Gale Moore)

User 4: female, 30, PhD student in Computer Science, used MSWord regularly, highly proficient with general computer applications (Joanna McGrenere)

4.2.3 Methodology

The process of installing the software, collecting and processing data, and modifying the participants' PIs is described below.

Prototype and Logger Install

A short meeting was booked with each participant during which the researcher installed both the prototype and the software logger on the participant's computer. These were briefly demonstrated. Participants were asked which menu items and which toolbar items they would like in their PI. They were encouraged to initially select only items that they used regularly. Unfortunately, the logger could not be set to work automatically but rather had to be operated by the user. The researcher explained the exact steps required to operate the logger; the participants were instructed to create one log file per day of usage, and were shown how the prototype would remind them to start and stop the logger (see Figures H-10 and H-11 in Appendix H). All participants were located on The University of Toronto campus and the researcher planned to be on campus for at least the first couple of days that each participant was using the prototype in case any problems occurred. If the researcher was not contacted by a participant within a couple of days after the install she sent an email to ensure that everything was okay.

Data Collection

The researcher met with each of the participants in their offices on a regular basis (when it was expected that they would have accumulated approximately 5 days worth of log files). Software logs were collected and a very informal interview was conducted. Participants were asked if they had any problems, what were their impressions of the new interface, if they would like any adjustments to their PI, and if they had the option to have the prototype removed and go back to the regular interface, would they choose to have it removed. These sessions with users were usually very brief. Beyond the initial congenial chit chat, talking about the prototype usually lasted for at most a couple of minutes. Because of the informal nature of these sessions, they were not audio taped but rather the researcher made extensive notes immediately following the sessions.

Growth of Personalized Interface

The researcher constructed the personalized interface during the regular sessions held with each participant as described above. Adding a function was very quick and easy to do. It only required modifying the appropriate interface file and did not require MSWord to be shutdown if it were open at the time. The goal was to simulate continuous interface construction, albeit at a coarse level of granularity.

Duration

Participants each used the prototype for 2 to 3 months during the summer of 2000. Because of summer vacations and conferences, however, there were periods during which the prototype was not used.

Processing Log Files

When the participants first began using the prototype it was not possible to determine their use of the different interfaces and the associated functions. This is because the log files generated by MSTracker were so dense, and they required software tools to extract the high-level interactions. These tools were built while the study was taking place. The tools were, however, sufficiently operational part way through the study to enable us to determine interface and function usage from that point onward.

4.3 Pilot Study – Key Findings

The key findings relate to both quantitative and qualitative data collected as well as to the methodology and tools used for the Pilot Study.

Log Data

Table 4-1 summarizes the data collected from the logs. A number of observations can be made about the data in that table:

- User 1 used MSWord relatively little in comparison with the other 3 participants. In fact User 1 had an unexpected temporary change in job description starting in August, which explains why there were no logs collected after August 1st.

- User 1 demonstrated a preference for the DI, User 2 for the MI, User 3 for her PI, and User 4 for both the MI and her PI. In the case of User 2, the MI and her PI contained almost identical functions.

- On average the participants only used 50% of the functions that they asked for in their PI.

- Apart from User 2, who on four occasions switched from the DI back to the MI, and User 3 who did this on two occasions, participants tended not to "switch down". Once they switched to an interface with more functionality in it, they did not switch back to an interface with less functionality. From reviewing the actual logs, we see that in some cases this was because of the continued need for functions only available in the fuller interfaces.

	User 1	**User 2**	**User 3**		**User 4**
Logs					
First log	June 29	July 10	July 29		July 11
Last log	Aug 1	Sept 20	Sept 25		Sept 7
# of logs collected	12	26	14*	10	21
Switching between interfaces					
Switched to PI in % of logs	25%	11.5%	64.3%	10%	57.1%
Switched to DI in % of logs	25%	42.3%	35.7%	30%	4.8%
Switched to MI in % of logs	--	15.4%	7.1%	--	--
Total # of switches to PI	5	4	17	1	13
Total # of switches to DI	7	13	8	5	1
Total # of switches to MI	--	4	2	--	--
Launches					
# of times MSWord was launched (*l*)**	20	96	17	28	24
% of *l* with switches to PI only	15.0%	--	17.6%	--	50.0%
% of *l* with switches to DI only	25.0%	7.3%	--	10.7%	4.1%
% of *l* with switches to PI & DI	5.0%	2.1%	29.4%	3.6%	--
% of *l* with switches to PI & MI	--	--	5.9%	--	--
% of *l* with switches to DI & MI	--	4.2%	--	--	--
% of *l* with no switch	55.0%	86.5%	47.1%	85.7%	45.8%
Time					
Total time logged in Word (h:m)	5:55	8:14	19:7		20:18
% of time in MI	11%	63%	13 %		31%
% of time in PI	9%	1%	68%		66%
% of time in DI	79%	36%	20%		3%
Functions					
# of functions chosen to be in PI ***	38	18	47		74
% of functions in PI that were used	36.8%	55.6%	70.2%		37.8%
Total # of functions used	26	30	52		43

* User 3 had 24 logs in total. For the first 14, MSWord launched with MI showing. For the remaining 10, it launched with PI showing.

** Note that a log represents the usage of MSWord across 1 day whereas a launch represents one sequence of starting and then stopping MSWord. One log can contain multiple launches.

*** Based on the counting metrics from Study One, there were 15 functions that were available in every interface (no choice available); these included vertical scroll, horizontal scroll, and interface toggle. The functions chosen by a participant were chosen in addition to the base 15 functions.

Table 4-1: Log data from the Pilot Study.

Concept of Multiple Interfaces

In general the concept of multiple interfaces was easily grasped by all participants, right from the start. At one point User 2 expressed a slight confusion. Upon returning from an extended vacation she asked "Why do I need that interface anyway?" when the researcher, as usual, asked if she would like any changes to her PI. This confusion over the purpose of having both the MI and PI was understandable – in User 2's case the two interfaces were virtually identical and the MI was being displayed on startup. For this participant it probably would have made more sense to have just the PI and the DI with the PI displayed on startup.

Content of PI

At the first meeting, each participant told the researcher which functions to include in her PI. Beyond this initial construction, however, there was surprisingly little adjustment to any of the participants' PIs. There were a few additions. There were no deletions. The modifications made for each participant are described below. In addition we note whether there were any regularly used functions that did not appear in a participant's PI. This provides an indicator of how well a participant's PI matched her usage needs.

User 1: No modification to PI.

There was no function outside her PI that appeared in more than 25% of her logs.

User 2: After 5 logs one menu item was added at the participant's request. Interestingly, this item was not used during the remainder of the Pilot Study. After 26 logs three more menu items were added.

When the researcher met with User 2 on the occasion that these 3 were added, the participant was initially asked if there was any adjustment that she would like. She identified 1 function without much hesitation. At this point in the study it was possible to get meaningful information from her logs, so she was asked about particular functions identified by the logs that she was using that were not in her PI. There were 12 such functions but she was only asked about 7 of them (those that had been used four or more times). Of these 7 she only wanted 2 of them – these 2 had been used in 5 and 7 logs respectively. The others had only been used in 1 or 2 logs and she didn't want them. She wasn't told by the researcher how often she had used them, but was just asked about whether or not she wanted the function itself. When asking her about the functions, she would sometimes look for supporting information, saying something like "No, I don't think I really used that very much did I?" The decision-making process described here points to the fact that frequently used functions were desired rather than recently used functions and that supporting information (such as usage information) could be useful when personalizing.

By the end of the study, there was no function outside her PI that appeared in more than 8% of her logs.

User 3: After 13 logs one button was added to the toolbar.

There was no function outside her PI that appeared in more than 4% of her logs.

User 4: No modification.

There was no function outside her PI that appeared in more than 10% of her logs.

In terms of adding functions to their PIs, the participants appeared to have a good sense of what they needed/wanted in their PIs. By the end of the study no participant made regular use of any function that was not available in her PI. If a participant expected to use a function in the future then she asked to have it added. Otherwise, she didn't want it, even if it had been used extensively in a particular day. For example, User 1 and User 2 both had instances when they used a function not in their PIs many times over a short period of time (1-3 days). When asked if they wanted any functions added to their PIs, they both identified this high-use activity to the researcher but both declined to have the used function added to their PI because they said that they rarely did this activity. This shows that an adaptive interface based on frequency and recency of use alone would not match what the user wants.

Interface Visible on Launch

All participants began with the MI visible on launch of MSWord. User 1 and User 3, however, both asked to have their PIs set to be visible on launch; User 1 asked after using the prototype for 6 days and User 3 after 12 days. Initially this request could not be accommodated because modifications to the code were required[26]. By the time these modifications were complete User 1 had used the prototype for 12 days and User 3 for 14 days. At this point the researcher gave User 2 the option of having MSWord launch into her PI and she declined because she was still predominantly using the MI.

Why did User 3 and User 1 want to startup with their PI? For User 3 there was one particular function that she used a lot that was not available in the MI and so she was regularly switching to her PI. User 1 simply indicated that she was not too bothered by the clutter and would rather start in her own interface.

User 4 preferred to start in MI because she felt it was often all that she needed. Her PI had significantly more functions in it than the MI and she preferred to have as few functions as were necessary.

Switching Between Interfaces

As indicated in Table 4-1, participants generally did not switch to an interface with less functionality than their current interface. User 2 was an exception. She switched from the DI to the MI on 4 occasions and even reported this to the researcher during one of the regular visits stating that "less is best". Given that she switched to the DI on 13 occasions this means that approximately one third of the times that she switched to the DI she switched back to the MI.

According to the quantitative data, User 3 exhibited similar behaviour. She switched from her PI to the MI on 2 occasions. However, given that she switched to her PI on 17 occasions and actually launched into her PI 28 times, switching down to the MI only occurred about 5% of the time. Furthermore, she reported in an interview that at one point she switched down to the MI because she felt she "should". So rather than being compelled to change interfaces, she consciously switched, likely thinking that it was in line with the research objectives.

[26] Initially, having the MI visible on launch was hard coded into the prototype. The prototype was changed part

User 1 and User 4 did not “switch down”.

An interesting observation is that Users 2, 3, and 4 expressed a preference for having unused functions tucked away and yet, with the exception of User 2, once they had “switched up” to an interface with more functions, they did not later feel compelled to “switch down” to an interface with fewer functions. There are a number of possible explanations for this, which include the fact that these users would still be more familiar with the DI from years of prior use and may not have remembered that they could switch to something with fewer functions. Or, it could be that they expected to need further functions that were only available in the “larger” interfaces and didn’t want to be continuously switching back and forth in a single word processing session. Finally, the behaviour could simply be explained by the fact that users could not be bothered to switch.

Logger Not Designed to be Used in a Field Study

All participants complained about the need to start and stop the logger, MSTracker. All participants freely admitted to not starting it on occasions when they were under time pressure and they had to “do a little something”, for example going in to a file to look at something or to print it. Midway through the study users were told that they could just keep MSTracker running all day if they wanted rather than starting it and stopping it every time they launched and closed MSWord. This, however, was not a solution without problems: it resulted in enormous log files which was a logistical problem and certainly has privacy implications.

Ideally, these “quick” accesses to MSWord should have been captured. It is precisely for these uses that hardly any functions were needed and so they shouldn’t be lost from the data collection. This also means that the row in Table 4-1 that indicates the percentage of launches with no interface switches is only a conservative estimate.

way through the Pilot Study so that interfaces would be listed lexicographically as described in Section 4.1.2.

Preference to Use Multiple Interfaces

Participants were asked on several occasions throughout the study whether if they had an option to stop using the multiple interfaces and to revert back to the regular interface for MSWord they would choose to do so. And at the end of the study they were given the option to keep the prototype.

- User 1: After 6 days of use when asked if she would prefer to have just the DI she said “No, it’s so easy to just click and get everything if you want it”. Again after 12 days of use she said that having these interfaces is not much different than just having the full interface – with the full interface “you just know what you need to use and you click on it.” When given the option to keep the prototype at the end of the Pilot Study she thought that it was best to uninstall it. In general this participant was indifferent to the prototype throughout the study.

- User 2: She was surprised at how little she used when she saw her personalized interface for the first time and said “Why do we need all that stuff anyway?” She mentioned early on that she finds it less confusing with fewer things “up there”. On a few occasions she said “less is best”. At the end of the study she said “Well if it is over I guess you might as well take it off.” When she was told that she could keep the prototype without using the logger she said “Well okay, I’ll keep it then!” (Note from Table 4-1 that this user had 96 launches of MSWord; for the majority of these she started and stopped the logger on a per launch basis. She associated the prototype and the logger as one unit and didn’t realize that they could be operated independently.)

- User 3: She had to uninstall the prototype for stability reasons (see next section) as the Pilot Study was just wrapping up. She expressed a preference for the multiple-interface prototype and said that it would be great if there were some way she could easily add functions to her PI herself.

- User 4: continued using the prototype.

Stability of Prototype and Logger

For 3 out of the 4 participants, the prototype installed easily and there were no technical problems reported. For one participant, User 3, there were difficulties installing the prototype and it was determined that the cause was a bad interaction with MSWorks[27] which came preloaded on the participant's computer. The participant wasn't actually using MSWorks and once it was uninstalled the prototype worked fine. Another problem was reported soon after the prototype had been installed by the same participant when she installed the software Adobe Acrobat[28] that also interacted with the MSWord interface by adding several menu items. This caused a mismatch between the actual structure of her menus and that which was recorded in her interface files. To solve this problem it was necessary to reconstruct her PI and MI interface files. It was clear that the prototype implementation was not sufficiently robust to handle changes to the MSWord interface that occurred after the prototype had been installed. We anticipated this problem and prevented users from customizing through the native customization facility (as described in Section 4.1.3), however, we could not prevent the problem from occurring when caused by the installation of external applications.

User 3 also reported her system crashing once, which we later suspected was related to the bizarre glitch described in Section 4.1.3. This participant had to disable the prototype at this point (by deleting the template file from the startup directory). None of the other participants experienced such a problem or disabled the prototype. Thus, apart from interactions with other software applications and one crash, the prototype proved to be very stable.

MSTracker also proved to be stable; there were no reports of it crashing. However, the researcher had some concerns that this logger was not collecting data accurately. It did not report interactions that did not occur, but seemed on occasion to miss some interactions that did occur. Only later was it verified that this was actually the case. (See Section 6.2.2 for a more detailed discussion of our difficulty with MSTracker.)

[27] MSWorks is a lightweight office application that has hooks into the MSWord interface.

[28] Adobe Acrobat is an application that creates and views files in Portable Document Format (PDF). PDF lets the user view and print a file exactly as the author designed it, without needing to have the same application or fonts used to create the file. When a user has Adobe Acrobat installed on his/her machine, MSWord allows the user to "print" documents in PDF format.

Logistical Issues of Running a Field Study

Not all participants had the required access privileges to install software on their own machine. For User 1 it was necessary to go through the system administrator to get MSWord upgraded to the Office 2000 version and to install MSTracker. Except for the time it took to talk and coordinate with the system administrator, this was not a problem. User 1 is part of the Department of Computer Science at the University of Toronto and so the system administrator was very open to making changes to User 1's machine for research purposes, with very few questions asked. We suspect that if we had had to deal with system administrators outside of our own department, it would have been significantly more challenging to get access to users' machines.

Collecting and processing the logs was very time consuming. When some of the users began leaving the logger running throughout the entire day (even when MSWord was not in use), the log files became enormous – as large as 5 MB. This meant that they could no longer be copied onto a floppy disk and had to be emailed from the participant to the researcher. The email server used by the researcher is set to receive email that is no larger than 10 MB which meant that multiple emails were often needed to send the log files.

During the first meeting with participants they were given verbal instructions on how to uninstall the prototype should it be necessary. In the one instance that the prototype was uninstalled by User 3, she contacted the researcher for a reminder on how to perform the uninstall. This was an indicator that more formal written instructions should have been given to the participants.

Summary of Findings

In general the Pilot Study was a success. It encouraged us to continue on our research path. Of our two unbiased participants, User 2 demonstrated a strong preference for having a reduced interface and found the multiple-interfaces design with the toggling mechanism to be intuitive. User 1 was indifferent but did not at all seem hindered by having the multiple interfaces. One could speculate that if she used MSWord as much as she had used it prior to the Pilot Study and did not associate the logger with the multiple interfaces, that she would have been less indifferent. On the other hand, regardless of amount of use, she may have remained indifferent. User 3 and User 4 had a preference for the multiple interfaces but this is tempered by the fact that both participants were biased with respect to the research objectives.

Recall Question One, the first of the two questions about personalization that was posed in Section 3.5: if one assumes an ideal personalized interface, one that includes only those functions the user would like with zero overhead to the user, does this interface have value? In the Pilot Study we only approached zero overhead; the participants had to manage MSTracker and had to meet occasionally with the researcher, which constituted some overhead. We were not able to assess all aspects of value, but rather used satisfaction as expressed by the participants to represent value. Leaving these nuances aside, it would appear that the answer to this question is not a simple yes or no. For some users, having a personalized interface appeared to be beneficial and for other users it was neutral in value.

Having said this, the MI was of questionable value. Users 1 and 3 largely ignored the MI. User 2 used the MI but would have used her PI had it been the interface visible on launch and in fact she experienced some confusion at least at one point over the differences between the MI and her PI. User 4 did make use of the MI. She had at least twice as many functions in her PI than any of the other 3 users and so the difference between the MI and the PI was significant in her case.

4.4 Towards Study Two

The results from our Pilot Study lead us to speculate on a number of interesting patterns of both user behaviour and user preference that could be explored in the context of a formal evaluation of the multiple-interfaces prototype.

Users want functions in their PI that they use regularly (daily, weekly, monthly) rather than recently.

Our findings show that even when users use a function many times in a particular day, they won’t choose to have it in their PI unless they expect to use it with some regularity in the future. Validating this finding would be interesting because it would indicate that an adaptive algorithm based on recency (such as the algorithm in MSWord 2000) is sub-optimal. It would be necessary to determine *how* sub-optimal the adaptive algorithm is in comparison to users selecting their own functions for their PI.

Novice users would feel a better sense of mastery and therefore more confident and more satisfied with multiple interfaces. Feature-shy users would, in general, feel more satisfied with multiple interfaces.

Novice users would likely stay in the MI or in their PI (which would also be small) because there would be fewer functions to contend with than in the DI and they would therefore be more likely to master the functions within one or both of these interfaces.

Recall that feature-shy users from Study One are those that do feel negatively impacted by the number of features and do not feel compelled to have the most up-to-date and complete software just for the sake of having it. For non-novice feature-shy users, they would have the option of having both an interface with few functions (MI) and one with just those functions they want (PI) rather than being in the DI all of the time. The ability to have just the functions they want would lead to greater satisfaction.

After completing the study, we had User 1 and User 2 fill out the Study One questionnaire: User 1 was clearly categorized as feature-keen and User 2 as feature-shy. This was an early indicator that the gains to be had with multiple interfaces in terms of affective response would be had by the feature-shy users.

Users would learn more about the core functions (those that are included in the minimal interface) with multiple interfaces.

Having multiple interfaces partitions the exploration space, giving users a reasonable space to start in, namely the MI. Users would feel less intimidated to explore about 20 functions and learn those functions than if they had 200 functions to explore. Users, however, *may* learn less about the functions outside of the MI than if they initially had all functions visible. By partitioning the exploration space there is a very distinct boundary created. Some users may feel that they cannot cross that boundary. On the other hand, they *may* actually learn more. Some users may feel more confident having mastered the "first-level" and feel compelled to try and master another level[29].

Bringing users into the "update loop" to allow them to participate in the construction of their own PI would result in an increased sense of mastery and satisfaction.

Participating in or controlling the interface construction would play a positive role as long as the cost of participating – the time required on the part of the user – is not too high. We speculate that the reason that the great majority of users do not use current customization facilities in MSWord is because the cost is too high. The user has to learn that the customization facility actually exists and then they have to learn how to use it, which is non-trivial.

Control over the interface is one of the core issues in the debate between adaptable and adaptive user interfaces. Although all of the patterns of user behaviour and preference mentioned above would be worthwhile pursuing, we focus on the last one, namely control, in our formal user study that is described in the next chapter.

[29] Users tend to want to be a master of a functionality subset rather than a novice of the full system (Section 2.3.1), but does this actually promote learning?

CHAPTER 5 Study Two – Removing the Wizard

Study Two directly followed the Pilot Study and focused on Question Two listed in Section 3.5, namely, how can the interface be designed such that it provides sufficient control to the user but it only requires minimal effort from the user? The Wizard of Oz component of the prototype used in the Pilot Study had to be removed and replaced with a facility that allows users to easily construct their own personal interfaces.

This interface design can be approached in two ways. The first is to have the personal interface constructed through some intelligent components that determine what features the user needs. In general, the goal of intelligent user interfaces is to assist the user by offloading some of the complexity. Adaptive interfaces, which are a form of intelligent interfaces, rely on intelligence to automatically adjust in a way that is expected to better suit the needs of each individual user. MSWord 2000 takes this approach with its adaptive interface. (See Appendix A for a detailed description of the MSWord 2000 adaptive interface.) By contrast, a second approach is to put the user in control of deciding what functions need to be added or removed from his/her personalized interface, in other words, provide a user-adaptable interface. There has been a debate in the user interface community between those who promote intelligence in the interface and those who promote "comprehensible, predictable, and controllable interfaces that give users the sense of power, mastery, control and accomplishment" [Shneiderman & Maes, 1997]. The issue is far from being resolved.

As we indicated at the very end of the previous chapter, our opinion has been that keeping the user in control provides a promising direction for design. We expected that having users participate in the construction of their own personalized interface would have a beneficial impact. This is not to say that intelligence can never play a role in the interface, but human capabilities had to be explored first and from there it would be possible to determine where the "sweet spots" might lie, where humans can be assisted in a way that is comprehensible and unobtrusive to them.

Findings from the Pilot Study showed that the multiple-interfaces design was a reasonable design alternative and perhaps preferable for some users to an all-in-one design such as MSWord 97 or an adaptive design such as MSWord 2000. But this hypothesis had yet to be tested. Our goal for Study Two was to complete the design of the multiple-interfaces prototype such that the Wizard of Oz methodology was no longer required for evaluation and such that the user was in total control of the personalization process. A paper that describes the design of Study Two and selected results has been accepted for publication to ACM CHI 2002 [McGrenere, Baecker, & Booth, 2002].

5.1 The Prototype

The design of the prototype for Study Two is a direct extension of the Pilot Study prototype. The following modifications were made:

- The minimal interface was removed. The only user from the Pilot Study who benefited from this interface was User 4 who had a relatively large personal interface. Although User 2 used the minimal interface, she would have used her personal interface if it had been the interface shown on startup. For users who use a large percentage of the functionality in MSWord, the minimal interface may play a significant role, however, Study Two was not going to focus on these users and so it was removed.

- The naming for the “Default Interface” was changed to the “Full Interface”, which reflects more accurately the contents of this interface.

- A mechanism that allows users to construct their own personal interface was added. This consists of the button “Modify <user name>’s Interface” that is located on the menubar. This mechanism permits both the addition of functions to and the deletion of functions from the personal interface. Upon invoking this button the user is given the option to **Add** or to **Delete**. When users select **Add**, the interface displays what appears to be the full interface from which they can select any toolbar item or menu item in the same way that it would be selected upon invocation. Once an item is selected a small dialog box pops up to confirm the selection. When users select **Delete**, the interface appears to be the personal interface from which any item can be selected. The deletion of an item is also confirmed with a dialog box.

Example screen captures of the revised prototype, including the process of adding and deleting, are shown in Appendix I.

The prototype was implemented in Visual Basic for Applications (see Section 4.1.2 and Appendix Q for details).

5.2 Field Study vs. Controlled Laboratory Experiment

Our first decision with respect to evaluating the prototype was the method to be used. Two common evaluation methods for user interface designs are field studies and laboratory experiments.

Evaluating personalizing behaviour has some inherent challenges. The extent to which a user personalizes and the nature of the personalizations that are made are likely to be very dependent on the user being aware of the functions that they currently use and expect to continue to use in the future, in other words, dependent on the user intimately knowing his/her future tasks. Based on this observation it seemed almost certain that evaluating personalization in the context of a field study would reveal more true-to-life user personalizing behaviour. A well-documented challenge in field studies, however, is the ability to make comparisons among different conditions. This is because the individual differences of users and their tasks and environments are too great to randomize unless a very large number of participants is used, which then becomes logistically infeasible. Thus, for a rigorous comparison of the impact of various interfaces (i.e., conditions), the evaluation should be carried out in the context of a controlled laboratory study.

As we have just seen, evaluation goals can compete in that one goal may necessitate one style of study and another goal, a different style study. In our case the goal of understanding personalization favoured a field study whereas the goal of comparing our prototype to other interfaces favoured a laboratory experiment.

Given the competing goals and methodologies, it was necessary to prioritize our goals for this study. Our priority is given by the order of the research questions given below:

1) How do users experience the novel aspects of the multiple-interfaces design? This includes understanding how users personalize and how they make use of the two interfaces.

2) How do the various interfaces compare with respect to user satisfaction, productivity, and other possible dependent variables?

Our study goals dictated that Study Two would be a field study in the form of a quasi-experimental design. Quasi-experimental designs are used in natural social settings where full experimental control is lacking [Campbell & Stanley, 1972]. This can be contrasted with experimental designs in which there is greater control. The ability to fully control or schedule experimental stimuli – to decide exactly when and to whom stimuli will be applied and the ability to randomize exposures – is what makes a true experiment possible.

Study Two received ethics approval for the use of human subjects from the Social Sciences and Humanities Review Committee at the University of Toronto.

5.3 Participants

Twenty experienced MSWord users participated in this study. Participation was solicited electronically through newsgroups and listservs that were broadcast across the University of Toronto campus and the surrounding city. The Call for Participation is provided in Appendix J. In order to participate users had to meet the following base criteria: they had to have been using MSWord 2000 for at least 1 month, had to do the majority of their word processing on a single computer, had to spend a minimum average of 3 hours word processing per week, had to have MSWord expertise above the novice level, had to be at least 20 years of age, and had to live at most a half hour's drive from campus. In order to ensure that these base criteria were satisfied, prospective participants completed an online screening questionnaire, the Microsoft Word Study Preliminary Questionnaire (Appendix K). Ninety-eight people completed this questionnaire and were considered in the order in which they applied.

A user's level of expertise was assessed with a Microsoft Office screening questionnaire[30] that was embedded as Section II of the Preliminary Questionnaire. The screening questionnaire categorizes expertise into five groups: novice, beginner, intermediate, advanced, and expert. A user had to be ranked at least as a beginner in order to participate in our study.

Personality differences with respect to heavily featured software were considered. We included 10 feature-keen participants and 10 feature-shy. A person is categorized as feature-keen, feature-neutral, or feature-shy based on his/her response to statements about: (1) having many functions in the interface, (2) the desire to have a complete version of MSWord (i.e., not a Light version), and (3) the desire to have an up-to-date version of MSWord. To assess these personality differences we re-used the statements from questions 33 and 34 from the questionnaire Experiencing Word Processing (Appendix F) that were used to construct the Feature Profile Scale (Appendix G) in Study One (Chapter 3). Those statements were replicated in Question 14 of the Preliminary Questionnaire used in Study Two[31].

In conducting research one generally hopes that the results found will generalize beyond simply those who participated in the study. The extent to which results will generalize is of course related to the extent to which the sample is representative of the true target population, in our case MSWord 2000 users. There was no sampling frame of this user population available to us and so we weren't able to achieve a simple random sample, i.e., a representative sample. We describe our sample because it suggests where limits to

[30] The Office Knowledge Test (Version 3) screening questionnaire is a validated instrument that is used internally at Microsoft for usability evaluations [Davis, Dye, Johnson, & Bell, 1999]. It is a proprietary instrument and for that reason a copy of it could not be included in this dissertation. Note that this questionnaire assesses expertise of the whole MSOffice product suite, which includes other applications in addition to MSWord. Although we were particularly interested in MSWord expertise in Study Two, Microsoft did not have an instrument to assess expertise of MSWord only. This is one limitation with our study.

[31] Of the 98 people who completed the Preliminary Questionnaire, 19 were feature-shy, 48 were feature-neutral, 30 were feature-keen, and 1 was not categorized because of failure to complete all of the statements in the questionnaire. Note that one might initially assume that this data implies that the true distribution of MSWord users across the Feature Profile Scale is different than that found in Study One (Section 3.3). It is possible, however, that feature-shy users are in general less inclined to participate in a study such as our Study Two and therefore fewer of these users completed the questionnaire. To accurately assess the distribution of users with respect to their perception of complex software one would have to have a large sample of users complete the statements independent of any other requirements (such as participating in a study).

generalizability may lie. An aggregate description of the participants is found in Table 5-1. Because of the relatively small number of participants, the participants are also described individually in Table 5-2.

	Gender		Mean age (/6)	Mean education (/7)	Mean MSOffice expertise (/5)	Mean # years using MSWord
	M	F				
Feature shy	3	7	2.9 = late 20s (1.1 SD)	6.1 (0.9 SD)	3.8 (1.2 SD)	7.1 (4.6 SD)
Feature keen	5	5	2.7 = late 20s (1.0 SD)	5.6 (1.3 SD)	4.5 (1.0 SD)	7.0 (3.2 SD)
TOTAL	8	12	2.8 = late 20s (1.0 SD)	5.9 (1.1 SD)	4.2 (1.1 SD)	7.0 (3.9 SD)

Table 5-1: Aggregate description of the 20 participants in Study Two (N=20).

The data show that there are differences between the feature-shy and feature-keen groups of participants in terms of gender distribution, education, and MSOffice expertise, but none of these differences are statistically significant. There are, however, a number of attributes of our sample that lead us to believe that it is not fully representative. On average, the participants appear to be highly educated (a rating of 5 equals the completion of an undergraduate degree) and long term users of MSWord. There are no secretaries or administrative assistants, roles that clearly include many users who do word processing, and based on Study One, include users of MSWord in particular. We do not have a definitive reason why we did not get participants in these roles. Perhaps the Call for Participation did not reach these groups or they were reached but individual users did not feel that they should participate in this study. For example, secretaries and administrative assistants typically do not "run" their own time at the office, so they may have felt that participation would not have been acceptable to their employer. Another likely point of difference between our sample and a representative sample is that graduate students make up one quarter of the participants. This is easily explained by the fact that the Call for Participation was sent to newsgroups on a university campus.

Achieving a fully representative sample is challenging and therefore rarely done in HCI research. It is not uncommon, for example, to have a sample consisting entirely of undergraduate students. Relative to this, our sample is certainly quite broad.

	User #	Occupation	Gender	Age	Education (/7)	MSOffice expertise (/5)	# years using MSWord
Feature shy	1	self employed, administrative computer tasks	F	20-29	5	4	6
	4	Software Developer	F	20-29	5	3	6
	6	Administrator	F	20-29	5	4	4
	7	Academic, Zoology	M	40-49	7	5	8
	11	Interactive Architect	F	30-39	6	5	11
	12	Interactive Architect	F	20-29	6	5	10
	13	self employed, media creation and course development	F	40-49	7	2	3
	14	Lawyer	M	50-59	7	5	17
	15	Graduate Student, Psychology	F	20-29	6	2	2
	21	Writer	M	30-39	7	3	4
Feature keen	2	Technical Support and Web Development	M	20-29	3	4	6
	3	Software Developer	M	20-29	5	5	6
	5	Graduate Student, Mechanical Industrial Engineering	F	20-29	6	4	4
	9	Graduate Student, Information Studies	M	30-39	6	2	3
	10	Graduate Student, Psychology	F	20-29	6	5	7
	16	Academic, Information Technology Management	F	30-39	7	5	14
	17	Academic, Medicine	M	50-59	7	5	8
	18	self employed, Web Designer	F	30-39	4	5	10
	20	Grad Student, Computer Science	M	20-29	6	5	5
	22	Teacher	F	30-39	6	5	8

Table 5-2: Individual description of 20 the participants (N=20).

Lost participants

Our goal was to have 20 participants – 10 feature-keen and 10 feature-shy. We began the study with 22 participants in the hopes that we would have at most a 10 percent dropout. These 22 began participating within an 8-day period. Within 2 weeks of starting, one feature-shy participant, User 8, virtually disappeared for reasons that were later determined to be family related. This prompted us to include one additional feature-keen participant and one additional feature-shy participant 3 weeks following the start of the study. One of the original feature-keen participants, User 19, was then disqualified because he was completing the questionnaires although he was not using the prototype (as shown by the logging data). The feature-keen participant who was added last, User 23, was disqualified at the end of her participation because the log files showed that the prototype had been deliberately tampered with (she claimed that it must have been done by her husband). This left us with 11 feature-shy participants and 10 feature-keen participants. In order to have the same number of

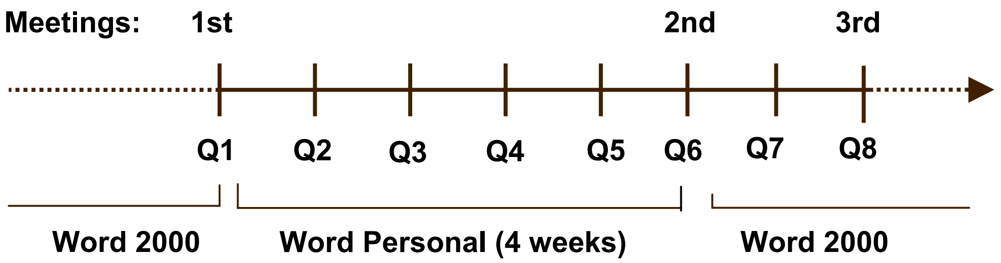


Figure 5-1: Timeline of the study protocol.

participants in the two groups, we chose to not include the data from the feature-shy participant who was added last, User 24. The decision to not include this particular participant was based on two factors: (1) she began the study 3 weeks after the other participants, and (2) relative to all other participants, she had done a large amount of customization to her installation of MSWord 2000 prior to her involvement in the study (by entirely reconstructing her **Standard** and **Formatting Toolbars**) and was therefore a clear outlier from the start.

5.4 Study Protocol

As previously mentioned, we chose to conduct a field study instead of a laboratory study because it was expected that true personalizing behaviour would be significantly more likely to occur with users doing their own tasks in their normal work context rather than in a lab setting with prescribed tasks.

Each participant was involved for approximately 6 weeks and met with the experimenter on three occasions as well as completed a series of short on-line questionnaires, Q1 – Q8, to assess experience with the software. Copies of Q1, Q2, Q7 and Q8 are found in Appendix N. Q3, Q4, Q5, and Q6 contain identical questions to Q2 and are therefore not included in the Appendix. Figure 5-1 provides a timeline for the study protocol.

First Meeting

The participant completed a printed version of Q1 that assessed the participant's experience of MSWord 2000. At the same time that the participant was filling in the questionnaire four programs were installed on the participant's machine – the prototype software which we called MSWord Personal[32], a software logger for capturing usage, a small script to transfer the log files to a backup server on the Internet, and a script to delete the prototype in the event of some technical malfunction. (Microsoft IV was the software logger used – see Section 6.2.4 for a description of IV.) Each participant's personal interface contained only six functions initially: **Exit** and the list of most-recently-used files in the **File** menu, **Open** and **Save** on the **Standard Toolbar** and **Font** and **Font Size** on the **Formatting Toolbar**.[33]

Q2 through Q6

These questionnaires assessed MSWord Personal. Q2 was completed within 2 days of the First Meeting and was intended to capture the participant's first impression of MSWord Personal. Q3, Q4, Q5 and Q6 were completed 1, 2, 3, and 4 weeks respectively from the First Meeting and were intended to capture the participant's experience of MSWord Personal over time.

Second Meeting

The Second Meeting was held within 1 day of Q6 being completed. MSWord Personal was uninstalled leaving the participant with MSWord 2000. The log files were collected on diskette.

Q7

Q7 assessed the participant's experience of MSWord 2000 one week following the Second Meeting. It was intended to capture the participant's reaction to returning to MSWord 2000.

Q8

Q8 asked the participant to rank MSWord 2000 and MSWord Personal in terms of each of the dependent measures 2 weeks following the Second Meeting.

[32] In order for participants to consider this prototype as a legitimate word processor we had to give it a legitimate name.

[33] These six functions were selected judgmentally.

Third Meeting

The Third Meeting was held within 1 day of Q8 being completed. The participant's machine was completely restored to the state it was in prior to the study. A final semi-structured debriefing interview was conducted with each participant. (The interview questions are given in Appendix O.) The log files were collected on diskette.

Instructions Given to the Participants

In advance of the study participants were only told that some changes would be made to their word processing software but they were not told the nature of the changes. At the First Meeting they were told that a new version of the software had been installed – MSWord Personal – which contained two interfaces. The experimenter toggled between the two interfaces once as a brief demonstration. It was pointed out that the personal interface contained very few functions initially but that functions could be added or deleted with the Modify button. The process of personalizing, however, was not demonstrated. Participants were told that there was no right or wrong way to use the interfaces and it was specifically mentioned that they could choose to use just one of the two interfaces and essentially ignore the other or they could switch between the interfaces in any way that suited their needs. Participants were not told that MSWord Personal would be uninstalled at the Second Meeting. The impression intended was that it would be used for the entire duration of the study. The exact instructions provided during the First and Second Meetings are given in Appendix L. Note that in addition to providing the instructions verbally, printouts of the text (as shown in Appendix L) were also given. No information about the goals or objectives of the study was provided to the participants at any time during the study.

Scheduling

In order to ensure the timely completion of questionnaires and meetings, an individual web page was constructed for each participant that contained all the necessary due dates as well as URLs to all the questionnaires. This acted as a shared resource between the researcher and each participant. (Appendix M shows an example of a participant's web page when the participant was part way through the study.) In addition, email reminders were sent by 9:00 AM on the due date of each questionnaire with the participant's web page URL directly embedded in the email, which facilitated quick access to the questionnaires. Reminders for each of the three meetings were sent one business day in advance. The participants' web pages were updated regularly to reflect completed activities. There was some flexibility in the scheduling in that if a participant was going to be unable to complete a questionnaire or attend a meeting on the scheduled date, the date could be adjusted slightly. Adjustments to the schedule were mostly made up front during the First Meeting.

Our goal with respect to scheduling was to avoid any confusion about the time and dates of meetings and the due dates of questionnaires. In general, we were very successful in that very few meetings had to be rescheduled during the study and similarly, there were few questionnaires that arrived late. See Section 5.8 for further details.

Compensation

Each participant received a $100 gift certificate for a local department store as compensation. In addition, there was a similar $100 gift certificate awarded as a prize to the participant who completed the most number of questionnaires on time.

Formal Design

The logistical constraints in conducting a field study precluded the counterbalancing of word processor conditions. The design is a 2 (personality types, between subjects) X 3 (levels, levels 1,3 = MSWord 2000, level 2 = MSWord Personal, within subjects) design where level 2 is nested with five repetitions.

The fact that the order of the conditions are not counterbalanced and that there is not a control group makes this a quasi-experimental design rather than an experimental design [Campbell & Stanley, 1972].

5.4.1 Pilot Test

The protocol outlined above was the final protocol used for Study Two. It was partially determined through our testing in the Pilot Study (Chapter 4), which was an in-depth early evaluation of the multiple-interfaces concept and our first implementation. It was also determined through another, albeit much quicker pilot test. For reasons of completeness, we briefly step back now and describe that pilot test.

The purpose of our pilot test was to ensure that the protocol for Study Two was basically sound and that all of the technology was working, including the prototype, the new software logger, the upload scripts, and the online questionnaires. The study duration was shortened in order to expedite this final stage of testing: 2 participants used the prototype for 2 weeks and then MSWord 2000 for 1 week.

Here we describe several aspects of the protocol used in the pilot test and the changes that were made to the protocol for the actual study. The changes discussed below are already reflected in the protocol as it is described in the previous section.

Scheduling

Pilot: Participants were provided with a printed schedule listing the due dates of the questionnaires and they were expected to complete the activities according to that schedule. If a scheduling change was required, they were expected to inform the researcher so that she could update her duplicate copy of the schedule. Participants were also allowed to schedule the time and date of the Second and Third Meetings early in the week that they were expected to occur. The pilot test showed that expecting participants to complete the questionnaires on time without any intervention was not realistic and that the "schedule-as-you-go" means of arranging the meetings resulted in excessive email exchange.

Study Two: The dates of all the questionnaires and meetings were scheduled with each participant during the First Meeting. Email reminders were sent on each questionnaire due date. One official online version of the schedule was maintained which eliminated the need for the researcher and the participant to coordinate updates to their respective copies of the schedule.

Duration of First Meeting

Pilot: During the First Meeting participants completed Q1 online and the researcher then installed the various required pieces of software. In addition, SR1, the latest service release for MSOffice, was installed as this was recommended for use with the software logger. The pilot test showed that Q1 took on the order of 15 minutes to complete, during which time the researcher essentially just waited. Participants engaged the researcher in conversation during the installation, which was distracting and caused the installation to proceed slowly. The installation of SR1 took approximately 30 minutes even though both pilot participants who needed the installation had high-speed Internet connections (the service release was downloaded from the Microsoft web site).

Study Two: Participants were given a printed version of Q1 to complete while the researcher performed the install. This occupied the participants and enabled the researcher to start the install process with little distraction. Participants were given instructions and asked to install the MSOffice service release on their own prior to the First Meeting[34].

Reliability of log file uploads

Pilot: The ftp upload process often did not complete properly from 1 of the 2 pilot participant's machines. It worked fine when it was installed but not thereafter. It was briefly investigated but the exact problem was never determined.

Study Two: The primary means for collecting the log data was by copying it to diskette during the Second and Third Meetings. The log file upload process was only used as a backup measure; in the event of a malfunction on a participant's machine that destroyed his/her log files, the upload guaranteed at least partial data.

5.5 Measures

The dependent measures were based on logging data and data collected from the eight questionnaires. From the logged data we extracted the total time spent word processing, the time spent in each interface, the number of toggles between interfaces, the trace of the modifications made to the personal interface, the trace of functions used, and summary statistics of function use. The on-line questionnaires included a number of self-reported

[34] During the initial stage of Study Two it became obvious that having users install the service release on their own was going to be problematic for a number of users. At this point the researcher double-checked with Microsoft that applying the service release was indeed a requirement (and not simply a recommendation) for using the software logger. Microsoft responded that the logger would work without SR1. The installation of the service release was therefore dropped from the Study Two protocol, which is why it is not reflected in Section 5.4.

measures. Each questionnaire presented the same series of 13 statements which were rated on a 5-point Likert scale (Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree). The statements are given below in the order in which they appear in the questionnaires:

[ease of use] This software is easy to use.

[menu control] I am in control of the contents of the menus and toolbars.

[learnability] I will be able to learn how to use all that is offered in this software.

[navigation] Navigating through the menus and toolbars is easy to do.

[engagement] This software is engaging.

[match needs] The contents of the menus and the toolbars match my needs.

[getting started] Getting started with this version of the software is easy.

[flexibility] This software is flexible.

[finding options] Finding the options that I want in the menus and toolbars is easy.

[control] It is easy to make the software do exactly what I want.

[discoverability] Discovering new features is easy.

[quickness] I get my word processing tasks done quickly with this software.

[satisfaction] This software is satisfying to use.

Q2 through Q6 also included statements specific to MSWord Personal. These were rated on the same Likert scale:

[personalizing mechanism easy to use] The mechanism for adding/deleting functions to/from my personal interface is easy to use.

[personalizing mechanism intuitive] This mechanism for adding/deleting functions to/from my personal interface is intuitive.

[personalizing mechanism flexible] This mechanism for adding/deleting functions to/from my personal interface is flexible – I can modify my personal interface so it is exactly how I want it.

[toggle easy to use] This mechanism for switching between interfaces is easy to use.

[concept easy to understand] This concept of having two interfaces is easy to understand.

[good idea] Having two interfaces is a good idea.

5.6 Hypotheses

The hypotheses below are categorized according to the two main goals of the study: (1) to evaluate the user's experience of the novel aspects of the multiple-interfaces design as it is implemented in MSWord Personal, including their personalizing behaviour; and (2) to compare our design to the adaptive interface of MSWord 2000.

5.6.1 Experience of Multiple-Interfaces Design

We hypothesized about the user's ability to construct an appropriate personal interface with the personalizing mechanism, the extent to which each of the two interfaces would be used, how the user would switch between the interfaces, the overall acceptance of the concept of having multiple interfaces, and how and why a user's personal interface would change over time.

H1 **Usage Hypothesis:** The majority of the participants will choose to use their personal interface – they will find the personalizing mechanism easy to use, intuitive, and flexible enough such that they will use the mechanism to include all frequently used functions and will spend the majority of their time in their personal interface. Given that the feature-shy are more bothered by excess functions, they will spend more time in their personal interface than the feature-keen.

H2 **Good Idea Hypothesis:** The concept of having two interfaces will be easily understood and will be considered a good idea. The feature-shy will find the idea to be more valuable than the feature-keen.

H3 **Approaches to Personalization Hypothesis:** Feature-shy users will be more likely to keep only frequently used functions in their personal interface and then switch to the full interface when an infrequently used function is needed. Feature-keen users will be more likely to have either all functions that they ever use in their personal interface or to largely ignore their personal interface and go directly to the full interface every time MSWord is launched.

H4 **Toggling Hypothesis:** The toggling mechanism will be considered an easy way to switch between the two interfaces. Because of the different approaches to using the two interfaces (H3), the amount the user toggles between interfaces will not be correlated with how valuable the user believes the idea of multiple interfaces to be.

H5 **Growth of Personal Interface Hypothesis:** Modifications to participants' personal interfaces will be dominated by additions and there will be relatively few deletions. At the outset there will be an initial period lasting at most 10 usage days (days in which word processing occurs) when a series of regular modifications will be made. The size of the personal interfaces will eventually reach a steady state and for the majority of the participants this steady state will occur by the end of the initial period.

H6 **Modification Triggers Hypothesis:** There will be a number of different triggers[35] that prompt participants to modify their personal interface. There will be an initial trigger to add functions because the personal interface will otherwise be almost unusable. Another trigger will be the immediate need of a function that was not initially added but that the user expects to use in the future. In addition, participants will likely add functions that they have used from the full interface on one or more occasion and that

[35] We use the term "trigger" to mean factors that influence users to modify their interface, as it was used by Mackay [1991]. See Section 2.4.3 and the discussion on page 123.

they expect to use again. Triggers for deletion will include the mistaken addition of a function and the desire to remove functions that are not being used.

5.6.2 Comparison to Adaptive Interface

Here we hypothesize about how the multiple-interfaces design of MSWord Personal compares to one particular instance of an adaptive design, namely MSWord 2000.

H7 **Satisfaction Hypothesis:** Feature-shy participants will be more satisfied with MSWord Personal than with MSWord 2000. The feature-shy will be more satisfied than the feature-keen with MSWord Personal.

H8 **Navigation Hypothesis:** Both feature-shy and feature-keen participants will feel that they are better able to navigate the menus and toolbars with MSWord Personal than with MSWord 2000.

H9 **Control Hypothesis:** Both feature-shy and feature-keen participants will feel a better sense of control with MSWord Personal than MSWord 2000.

H10 **Learnability Hypothesis:** Feature-shy participants will feel that they are better able to learn the available features with MSWord Personal than with MSWord 2000.

H11 **Three-way Comparison Hypothesis:** When asked to compare their overall preference for MSWord Personal, MSWord 2000, and MSWord 2000 with the adaptive menus turned off (standard all-in-one interface), feature-shy participants will prefer the multiple-interfaces design to an all-in-one design but will prefer all-in-one to adaptive. Feature-keen will prefer all-in-one to both the adaptive and multiple-interfaces designs.

5.7 Results

For technical reasons we are missing most of the logging data for one of the participants, User 9. We have his interface toggling and personal interface modification data but we do not have his function usage data (see Section 6.2.4 for an explanation of the different sources of logging generated in Study Two). For some undetermined reason, the IV logger that we used in Study Two stopped working on his system after it was initially installed and checked. Where this is relevant we note N=19 and otherwise N=20 can be assumed.

The analysis found in Section 5.7.1, which relates to the user's experience of our design, includes descriptive statistics and summarization of the qualitative questionnaire and interview data. The analysis found in Section 5.7.2, which relates to the comparison between our design and the adaptive design in Word 2000, includes both inferential and descriptive statistics as well as summarization of the qualitative data. We follow standard conventions for significance levels: findings that are $p < .05$ are deemed to be significant and findings of $.05 \leq p \leq .10$ are borderline significant.

5.7.1 Experience of Multiple-Interfaces Design

H1 Usage Hypothesis

The majority of participants did find the personalizing mechanism sufficiently easy to use, intuitive, and flexible such that they added all of their frequently used functions and spent the majority of their time in their personal interfaces.

Questionnaire data indicated that participants found the personalizing mechanism easy to use, intuitive, and flexible. These had mean ratings out of 5 of 4.3, 4.1, and 4.0, respectively. We had in fact expected the flexibility ratings to be slightly lower (in the range of 2.5 to 3.5, where 3.0 is a Neutral rating) simply because the personalizing mechanism did not allow for positioning functions in the personal interface. Although some users did articulate preferences for additional flexibility (see the section below on Design Suggestions), the quantitative data shows that the mechanism was sufficiently flexible for most participants.

During the 4 weeks that MSWord Personal was used, 14 out of 19 participants spent 50% or more of their word processing time in their personal interface (Table 5-3) and these same participants added all frequently used functions (those functions used on at least half of the days that the word processor was used).

	% of time in Personal Interface				Total
	≥ 0%	≥ 25%	≥ 50%	≥ 75%	
Feature shy	2	2	2	4	10
Feature keen	1		3	5	9
Total	3	2	5	9	19

Table 5-3: Number of participants by amount of time spent in their personal interface (N=19).

Counter to our expectations, feature-shy participants did not on average spend more time in their personal interface than did the feature-keen. The actual mean percentage of time in the personal interface was slightly lower for feature-shy (m=64%, SD=.33) than for feature-keen (m=74%, SD=.31).

If we consider all 19 participants we see that the great majority of functions that were used even 25% of the time or more were added to the personal interface (Table 5-4). For example, 18 participants each had one or more functions that were used on 25% to 50% of the days that word processing occurred and on average participants added 90% of these functions. We can see that the more frequently a function was used, the more likely it was added to the personal interface. Thus participants were indeed adding functions according to their usage.

	Frequency of Function Use			
	0% < and < 25%	25% ≤ and < 50%	50% ≤ and < 75%	75% ≤ and ≤ 100%
Feature shy	10 (59%)	10 (88%)	9 (94%)	5 (100%)
Feature keen	9 (73%)	8 (92%)	6 (100%)	1 (100%)
Total	19 (66%)	18 (90%)	15 (96%)	6 (100%)

Table 5-4: Number of participants who have at least one function used with the given regularity. The number in brackets gives the mean percentage of these functions that were added to the personal interface (N=19).

Table 5-5 provides some context for Table 5-4 by showing the aggregate number of functions that were used with the given frequencies. For example, if we look just at the group of functions that were used in 25% to 50% of the daily logs, feature-shy participants used on average 6.8 such functions. Combining the data from Tables 5-4 and 5-5 tells us that on average 88% of the 6.8 functions were added to the feature-shy's personal interfaces. The data shows that participants have more functions in their repertoires that are used infrequently than ones that are used frequently, which is not at all surprising. Frequently used functions are indeed quite rare – participants had approximately 1 function that they used 75% of the time or more and approximately 3 functions they used between 50% and 75% of the time.

	Mean Number of Functions Used with Given Frequency			
	0% < and < 25%	25% ≤ and < 50%	50% ≤ and < 75%	75% ≤ and ≤ 100%
Feature shy	31.20 (11.48 SD)	6.80 (1.75 SD)	3.80 (3.68 SD)	1.10 (1.60 SD)
Feature keen	28.22 (12.49 SD)	7.67 (6.20 SD)	2.44 (3.05 SD)	.33 (1.00 SD)
Total	29.79 (11.73 SD)	7.21 (4.34 SD)	3.16 (3.37 SD)	.74 (1.37 SD)

Table 5-5: Mean number of functions used with a given frequency (N=19).

Together this data indicates that participants were capable of personalizing according to their individual function usage, it was easy enough to do, and the personal interface was actively used. The feature-shy participants, however, did not make greater use of their personal interface than the feature-keen.

H2 Good Idea Hypothesis

The concept of having two interfaces was easily understood (mean rating 4.4 out of 5) but the claim that having two interfaces is a good idea only received a mean rating of 3.9. We discovered through the debriefing interview that the desirability of having two interfaces could be interpreted in two ways, namely that having both a personal interface and the full interface is better than (1) having just one interface with everything (i.e., a full interface), or (2) having just a personal interface. We had intended the first meaning. Without being able to identify how each user interpreted the statement, it is impossible to interpret the results.

There was no significant difference in the ratings of the multiple-interfaces idea between the feature-shy participants (m=4.0, SD=.73) and the feature-keen (m=3.7, SD=.75) (N=20).

H3 Approach to Using Two Interfaces

Participants were not told how they should use the two interfaces in MSWord Personal and were therefore able to approach it in any way that met their needs. Analysis of the log data and the debriefing interviews allows us to approximately discern the general approach participants took to constructing their personal interfaces. In general, the approach can be broken down into two components: (1) which functions were added, namely, only the most *frequently-used* functions or *all* used functions, and (2) when functions were added, namely, *upfront* within the first few days of usage or gradually as functions were needed (*as-you-go*). The results are summarized in Table 5-6.

Approach to Personalization		Feature shy	Feature keen	Total
	frequently-used, upfront	2		2
	frequently-used, as-you-go	1	3	4
	all, upfront		4	4
	all, as-you-go	3		3
	frequently-used, as-you-go, gave up	1		1
	all, as-you-go, gave up	2	3	5
	none, gave up	1		1
	Total	10	10	20
Summary				
Which functions?	Frequently-used	4	3	7
	all	5	7	12
	none	1		1
	Total	10	10	20
Added when?	Upfront	2	4	6
	as-you-go	7	6	13
	none	1		1
	Total	10	10	20
Gave up?	Gave up	4	3	7

Table 5-6: Desired approach to personalization. Terminology as follows:
***frequently-used** = add frequently-used functions only; **all** = add all functions that are used;*
***upfront** = add majority of functions right away; **as-you-go** = add functions as they are needed;*
***none** = user did not personalize; **gave up** = abandoned desired approach.*

In terms of deciding which functions to add, we see that adding all functions dominated 12 participants to 7. Participants who added all used functions generally expected to use their personal interface exclusively whereas those who added only regularly used functions expected to switch to the full interface when irregularly used functions were needed. In terms of when the functions were added, 6 participants took the approach of adding the great majority of functions that they expected to add upfront, and then only rarely adding additional functions. By contrast, 13 participants took the approach of adding some functions in the beginning and then gradually adding additional functions (as-you-go).

Seven participants *gave up* on their desired approach to some degree and for most this meant that they stopped personalizing, or only added very few functions beyond their first few days of using MSWord Personal. These participants used their personal interface to the extent that they could but would then switch to the full interface when a function not available in their personal interface was needed rather than taking the time to add it. For example, User 1 is categorized in “all, as-you-go, gave up”. She wanted to have all the functions she used in her personal interface but in the end she realized she was using a lot more functions than she expected, some of which she was learning for a new contract she started during the study. She did continue to personalize throughout the study but ended up just adding the most frequently used functions, which was not her desired approach. More typical behaviour of participants who gave up is described here by Users 3, 10 and 13 who are all categorized in "all, as-you-go, gave up":

> *I would start out of the personal. At the beginning I was adding things pretty regularly to the personal but I felt that I was just continually adding things and so eventually I would just start out and use the personal as long as it was convenient and then I would just switch once I felt like I needed to add another function. [Interview, User 10]*

> *So I would use that [her personal interface] when I was doing straight word processing but what I found is that whenever I was doing really complicated editing, and there is one project that I am working on where it is quite complicated, that I would get frustrated and I would switch to the full interface. [Interview, User 13]*

> *I'd switch back to the normal Word view [full interface] if I needed to do things like, if I had to do all the Word styles and formatting stuff it might be too many menu items to add so I wouldn't think that far ahead – I'd be too lazy to add like 58 different things so I'd just switch to the full view. [Interview, User 3]*

User 14 was categorized as "none, gave up" because he used the full interface almost exclusively and clearly wasn't willing to spend the time to explore his personal interface. It was obvious from his comments in the debriefing interview that he really did not understand the concept of a personal interface and by the time of the interview he had completely forgotten that he himself had added four functions to his personal interface during the First Meeting:

> *It wasn't worth the time for me to try to figure out what I wanted, what buttons I wanted. I've got so many things to do here without trying to figure out…* ***Did it intrigue you at all or was it purely a time thing? Even if you had time and were interested in the software would you have tried it out?*** *I think if someone had sort of taken me through for 20 minutes or half an hour and said – here are the attributes of it and then maybe I would have been interested. Without really knowing what it was going to do for me, one thing I really don't have patience*

> *for is playing around with the computer and trying to install things and uninstall things. I just don't have patience for that. [Interview, User 14]*

It is interesting to note that none of the participants who took the approach of adding functions upfront ended up giving up. This seems to suggest that adding upfront may be a more effective strategy than adding as-you-go.

In terms of the personality types of the participants, feature-shy participants were not more likely to keep only frequently used functions in their personal interface (4 participants) compared to all used functions (5 participants) as had been hypothesized. Feature-keen participants, on the other hand, were more likely to want all functions that they ever use in their personal interface (7 participants) rather than just the ones used frequently (3 participants). Feature-shy participants were more likely to use the add as-you-go strategy (7 participants) than the add upfront strategy (2 participants). Feature-keen participants favoured the add as-you-go strategy (6 participants to 4) but not as decisively as the feature-shy.

H4 Toggling Hypothesis

The toggling mechanism had a mean rating of 4.5 out of 5 for its ease of use, indicating that a toggle is an effective way to switch between more than one interface. There was no correlation between the total number of toggles and the rating of the two-interface idea, although, as previously mentioned, there were problems with the interpretation of the idea statement. However, if we compare the percentage of usage days in which there was at least one toggle to the full interface against participants' final ranking of the interfaces (H11), we see that those 13 participants who ranked Personal first only toggled to the full interface on average 27% (SD=.24) of the days whereas the other 7 participants did so on 60% (SD=.29) of the days, thus twice as often. Therefore it does appear that the extent to which MSWord Personal is valued is related to the amount of toggling.

H5 Growth of Personal Interface Hypothesis

This hypothesis is essentially saying that users would not be personalizing constantly and that the great majority of personalizing activity would occur at the beginning of usage (within the first 10 days of usage).

The data shows that the size of participants' personal interfaces, with the exception of seven deleted functions, did increase monotonically. All participants added functions and only 3 participants deleted a combined total of seven functions.

For the purposes of understanding how personal interfaces evolved, we direct our attention exclusively to the 13 participants who did not give up their desired approach to personalization (H3) as these participants set up their personal interfaces in a way that met their needs. This group includes User 9 for whom we only have partial logging data and so we omit his data from this analysis, which leaves 12 participants. For this group of participants there was an initial period when modifications were made regularly. This series can be defined by a first modification followed by subsequent modifications that were at most 2 usage days from the previous modification. Table 5-7 shows that the average duration of this initial period lasted 2.8 days, and for the participant who had the longest initial period it only lasted 5 days, which is well under the 10 usage days we expected would be the maximum. Of the total number of functions that these participants added during the 4 weeks, on average each had added 82% of his/her total by the end of the initial period.

On average participants personalized 3.8 of the days that word processing occurred, with the participant who personalized most frequently doing so on 6 days. The size of participants' personal interfaces did approximately reach steady state in that there was a point at which the size of the personal interfaces did not increase/decrease by more/less than 10%. We expected this point to be within the initial period for the majority of the 12 participants. The results show, however, that the steady state point was reached by the end of the initial period for only half of 12 participants (for 2 participants steady state was equal to the initial period and for 4 it was less than the initial period). On average this steady state period came at 4.75 days, which was on average 31% of the way through the 4 weeks in which MSWord Personal was used. On average the last modification came 75% through the 4 weeks.

	Minimum	Maximum	Mean	SD
Last day of initial period	1	5	2.75	1.22
Percentage of functions added by end of initial period	44.00	100.00	81.58	19.30
Number of days that participant personalized	3	6	3.83	1.11
Total number of days MSWord was used	6	26	16.50	5.25
Day at which 90% of functions have been added	1	13	4.75	3.70
Percentage of 4 weeks at which 90% have been added	5.00	87.00	31.32	26.21
Percentage of 4 weeks at which last personalization done	15.00	100.00	74.65	24.75

Table 5-7: Aggregate personalizing data (N=12).

The data can also be viewed on a day-by-day basis. Table 5-8 shows that within the first 2 days that MSWord Personal was used 81% of all of the functions that would be added had been added by the 12 participants.

Usage Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15+
N	12	12	12	12	12	12	11	11	11	11	11	11	10	9	9
Minimum	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Maximum	71	44	8	10	7	9	0	14	0	1	0	2	7	1	2
Total	315	80	11	12	12	15	0	17	0	2	0	3	13	1	4
Mean	26.25	6.67	0.92	1.00	1.00	1.25	0.00	1.55	0.00	0.18	0.00	0.27	1.30	0.11	0.44
SD	21.77	12.57	2.27	2.86	2.09	2.70	0.00	4.23	0.00	0.40	0.00	0.65	2.41	0.33	0.88
Cumulative Total	315	395	406	418	430	445	445	462	462	464	464	467	480	481	485
Cumulative % of Total (485)	64.95	81.44	83.71	86.19	88.66	91.75	91.75	95.26	95.26	95.67	95.67	96.29	98.97	99.18	100

Table 5-8: Day-by-day number of functions added to the personal interface (N=12).

Therefore the bulk of personalization did occur within the first few days of usage, and participants had added on average 90% or more of the functions that they were going to add by a third of the way through the 4 weeks. But participants were on average not finished personalizing until almost three quarters of the way through. This may beg the question of whether they were finished at all, however, the relatively few modifications made in the latter days suggests that even if participants had continued to use MSWord Personal beyond the 4 weeks, the personalizing activity would have been minimal.

H6 Modification Triggers Hypothesis

We begin our discussion with some definitions. We are using the term *trigger* to refer to a situation that causes the user to initiate changes in the personal interface by adding or deleting a function.

We listed three possible triggers for addition of functions in our modification triggers hypothesis:

Initial trigger: The desire to add functions when first using MSWord Personal.

Immediate-need trigger: The need to use a function that is not currently in the personal interface.

Future-use trigger: The expectation to use a function in the future.

Two triggers for deletion are given in the hypothesis:

Mistaken-addition trigger: A function was added by mistake.

Non-use trigger: A function is not being used.

As mentioned in the discussion of H5, there was very little deletion and so we briefly cover this behaviour first. Only 3 participants deleted a combined total of seven functions: one, two and four functions respectively. In all 3 cases the functions were deleted directly after they were added. When queried in the debriefing interview the participant who had deleted four functions indicated that she had been testing the personalizing mechanism, and the other 2 participants said that the deleted functions had initially been added by mistake. Thus, the mistaken-addition trigger applied to the deletion of three functions and the non-use trigger did not apply to any deleted functions. One might initially assume that the lack of the non-use trigger is explainable by the fact that MSWord Personal was only used for 1 month. To counter this, however, the great majority of participants indicated in the debriefing interview that they would not likely have bothered to delete functions even if they were not being used.

As in H5, in the remainder of this analysis we focus exclusively on the data captured from the 12 participants who did not give up on their desired approach to personalization (H3).

Isolating specific triggers for addition proved to be difficult. It is clear that there was some initial trigger in that 81% of the functions were added within the first 2 days (H5), but even within these days functions were sometimes added and then used immediately, suggesting that they weren't added within the scope of a "bulk initial add". A general observation is that participants tended to add multiple functions at the same time. The design of the personalizing mechanism is such that once it is entered the user can add/delete a single function or many functions, one after the other. In the debriefing interview a number of the participants articulated that when they were in the personalizing mechanism adding a needed function, they often added "related" functions. For example:

> *As soon as I needed something that wasn't on my personal interface then I went to the regular one and just added it and if it was, say the* ***Left Align****, then I just automatically added the* ***Center*** *and the* ***Right*** *just so that I would have all three there. [Interview, User 1]*

The data shows that participants added on average 6.7 functions each time they entered the personalizing mechanism (SD=4.95, N=12).

Our three triggers for addition proved to be too vague when it came time to categorize each added function by a single trigger. So rather than attempting to verify the hypothesis as it was stated initially, we began simply by identifying the patterns of usage with respect to addition, in particular, whether a function was used before/after it was added and, if so, how soon was it used before/after if was added. We have identified seven positions of usage relative to the moment at which a function was added. The naming convention is 'P' for previous, 'C' for current, and 'A' for after:

P: function was used on a day or days previous to the current day

CP: function was used previously in the current day

CP1: function was used directly before it was added (current day, one function call ago)

CA1: function was used directly after it was added (current day, next function call)

CA: function was used later in the current day

A: function was used some day or days following the current day

None: function was never used during the 4 weeks that MSWord Personal was used – 47% of the functions added (229 of 485) fell into this category.

Note that None is mutually exclusive to the others, CA1 is a subset of CA as is CP1 of CP. Otherwise, these positions of usage can be aggregated into patterns of usage in that a function can be used multiple times and so more than one usage position may apply for a given function. Table 5-9 shows a breakdown of the various patterns of usage that were observed. For example, the row with the pattern {P, CA, A} shows that 2 functions were added such that they had been used on previous days (P), they were used subsequently on the day that they were added (CA), and they were then used on one or more days after the day that they were added (A). In both cases, the function was used directly after it was added (CA1). Of the 485 functions that were added during the study only 0.41% satisfied the pattern {P, CA, A}. Given that both occurrences of CA in this pattern were CA1s, the final column shows 100 per cent.

Pattern of Usage	Occurrences	CA1 / CP1	% of Total	% of Total CA1/CP1
{P, CP, CA, A}	0	-	0	-
{P, CP, CA}	0	-	0	-
{P, CP, A}	0	-	0	-
{P, CP}	0	-	0	-
{P, CA, A}	2	2	0.41	100.00
{P, CA}	3	3	0.62	100.00
{P, A}	0	-	0	-
{P}	0	-	0	-
{CP, CA, A}	0	-	0	-
{CP, CA}	0	-	0	-
{CP, A}	4	1	0.82	25.00
{CP}	4	1	0.82	25.00
{CA, A}	57	28	11.75	49.12
{CA}	21	10	4.33	47.62
{A}	165	-	34.02	-
{}	229	-	47.22	-
TOTAL	485	45	100.00	

Table 5-9: Occurrences of patterns of usage (N=12).

Table 5-10 shows the aggregate number of occurrences of each of the positions of usage (e.g., the occurrences of the 'P' position include all patterns from Table 5-9 that include a P). We show the CA1s and CP1s separate from the CA and CPs respectively to illustrate the extent to which a function was used *directly* before/after it was added. We can see clearly that the usage positions occurring after a function was added (CA, CA1, and A) occur significantly more often than usage positions before (P, CP, and CP1), indicating that participants were much more likely to add a function before it was used than after it had already been used.

Position of Usage	Occurrences	% of 485 Additions
P	5	1.03
CP but not CP1	6	1.24
CP1	2	0.41
CA but not CA1	40	8.25
CA1	43	8.87
A	228	47.01
None	229	47.22

Table 5-10: Total occurrences of each position of usage and the percentage of the usage patterns for the 485 additions that included each usage position. Because a pattern can include multiple usage positions, the final column does not sum to 100 percent (N=12).

We can try to determine why a participant entered the personalizing mechanism on each of the occasions that it was entered. (Recall that there were often multiple functions added each time it was entered.) For each time the personalizing mechanism was entered we basically want to know was there a *dominant* function among those added. By listening to what our users told us in the debriefing interview and by looking at the raw data we developed a dominance model based on the most proximate usage of a function relative to the time it was added: usage in the current day dominates usage in the previous or future days, and immediate usage dominates usage in the current day that is not immediate. Thus CA1 and CP1 dominate CA and CP, and the latter dominate A and P[36]. We can now define dominant function more precisely: among the set of functions added at the same time, the dominant

[36] There were no occurrences of CA1 and CP1 in the same pattern and so they can be given equal ranking. Similarly, A and P can be ranked equally because they do not occur together except with CA which outranks both of them.

function is determined by looking at the set of corresponding usage patterns and selecting the usage position that dominates all of the usage positions in the set of usage patterns as determined by our dominance model. Table 5-11 shows that in almost half (47%) of the 91 times that the personalizing mechanism was used to add functions, a function was added that was then used immediately. This can be compared to only 2.2% of the additions of a given function occurring directly following the use of that function (CP1). In general there is a much higher likelihood that functions are being added before they are being used.

Dominant Position of Usage	Occurrences	% of Total
CA1	43	47.25
CP1	2	2.20
CA	11	12.09
CP	3	3.30
A	26	28.57
None	6	6.59
TOTAL	91	100.00

Table 5-11: Occurrences of dominant position of usage for each of the 91 times the personalizing mechanism was entered for the purposes of adding functions. (N=12)

We can now return to our original three triggers for addition defined at the outset of this section. Our analysis of usage patterns enables us to define these triggers more precisely and add a fourth trigger.

> **Initial trigger:** The desire to add functions when first using MSWord Personal. Any function added within the first 2 days of usage that does not include a CA1 in its usage pattern satisfies this trigger[37].
>
> **Immediate-need trigger:** The user has an immediate need to use a function that is not currently in his/her personal interface and therefore adds it. Any function that has a CA1 in its usage pattern satisfies this trigger.

[37] There is no perfect way to define the initial trigger. As previously mentioned, aproximately 81% of the functions were added in the first two days which strongly suggests some form of an initial trigger. One could argue that all those functions should be considered in the initial trigger category. The fact that some of these functions include a CA1 in their usage pattern suggests that not all functions added on the first two days are added in the scope of a bulk initial add and favours the more conservative definition we have given.

Previously-used trigger: The user has already used a function and expects to use it in the future. Any function added beyond the first 2 days of usage that includes P, CP, or CP1 in its usage pattern but does not include CA1 satisfies this trigger.

Future-use trigger: The user expects to use a function in the future and so adds it to the personal interface. Any function that does not satisfy any of the first three triggers satisfies this trigger.

Triggers	Number of functions that satisfy trigger	% of Total
Initial	372	76.70
Immediate-need	43	8.87
Previously-used	6	1.24
Future-use	64	13.20
TOTAL	485	100.00

Table 5-12: Triggers for adding functions to the personal interface.

Table 5-12 summarizes our findings with respect to triggers for the addition of functions. We see that the initial trigger and the immediate-need trigger together accounted for almost 86% of the functions added. The previously-used and expected future-use triggers accounted for the addition of the remaining functions which were typically added while an immediately-needed function was being added.

Additional Qualitative Feedback

This section is devoted entirely to what the participants had to say about MSWord Personal, in both the open-ended sections in questionnaires Q1 through Q8 and in the final debriefing interview. The goal is to bring the quantitative data to life by placing it in the context of the qualitative data. We include comments about the general multiple-interfaces design as well as specific design suggestions for improving MSWord Personal.

General Feedback

Having the full interface was generally well liked. While there were 2 participants who did not make use of it at all, the great majority did use it at least once and appreciated having it available:

> *I'd always want to have the full interface to go to just as like a baseline kind of thing. … Because that Word [the full interface] was there it was this safety net of – yah I know it's over there if I ever do need it anyways. [Interview, User 11]*

> *I like the security blanket of having the full interface but over a longer period of time I probably would have extinguished my use of it, pretty much. [Interview, User 17]*

> *What I would really hate is if the personal one – if you couldn't go back and forth. The fact that you could back and forth and that it was so easy to go back and forth, that was very good. [Interview, User 13]*

Some participants found that the time required to look through menu items and toolbar icons was reduced while they used their personal interfaces:

> *I really like having only the tools I use very frequently on my interface if I so choose. It makes me more efficient as I don't have to look around for the function I need. [Q2, User 15].*

> *While I use a standard set of features for most of my work, I am pleasantly surprised when I go to use a feature I haven't used for a while and find it's the only one in the menu. It makes my task faster and less frustrating. [Q5, User 12]*

> *I'd be in the Microsoft Word interface and it'd be like – oh God just too many buttons. Like I don't think that Microsoft does a really good job of making their icon match what the button actually does. And I will sit there and I will have to hover over the button and wait for the explanation to come up. And it's like oh man, what a waste of time! So that's when I'd find myself getting like, okay, I don't need all this crap right now. It's too hard to find things on all the menubars and that's when I'd switch back to the personal. [Interview, User 11]*

The cost of having a personal interface was the time to set it up and some participants specifically noted this:

> *In the startup phase I am adding one or two functions at a time which is a bit slow. [Q2, User 17]*

> *The personal interface is an interesting concept but seems time consuming to completely set up. I am still adding features to it. [Q3, User 18]*

> *It was a bit of a pain to spend time setting up the Personal version, but once you got it set up it was fine. It's just an investment of time at the beginning, or whenever you decide to personalize. [Q8, User 11]*

> *Initial configuration was time-consuming but it is ok if it only must be done once. [Q8, User 16]*

For some, having a personal interface just added more to the interface without sufficient benefit. The personal interface had to be managed in that it had to be constructed and the user had to switch between it and the full interface:

> *I use Word for a variety of applications, not all on a daily basis. I found that while I could set up a Word Personal for my daily writing requirements, when a different application [task] came up, I would: a) discover that the function was not currently in my personal interface, (b) switch between interfaces, (c) find the function within the full interface then, (d) spend time deciding to either stay in the full interface or figure out how to add the particular function to my personal interface. Ultimately I found the additive process of developing a personal interface somewhat frustrating. [Q8, User 21]*

> *I use a wide enough variety of functions in Word that I was always adding things to my personal interface, or switching to the regular interface, or forgetting that I was in my personalized interface and getting annoyed that I couldn't find a function I thought should be there. The regular interface may be more cluttered, but I prefer having all those functions right at my fingertips where I can quickly browse through and find whatever I need. [Q8, User 10]*

> *The thing is for me was that I want my software to be pretty much invisible to me and the personal required more visibility than I would have liked it. [Interview, User 13]*

For others, the benefit was substantial:

> *I think something like that [MSWord Personal] should be made available. It's a nice thing – it's a nice interface. I mean, you know, I don't know how easy it would be to be available to many people. I guess that you would have to package it or whatever. But it was a nice addition. I actually enjoyed it. [Interview, User 15]*

> *I think that Word XP*[38] *needs a personal edition even more than 2000! [Q8, User 17]*

> *I would like to have Personal re-enabled on my machine! [Q8, User 16]*

Design Suggestions

Although the personalizing mechanism in MSWord Personal was reported on the questionnaires to be easy to use, intuitive, and flexible, 3 participants commented about it being somewhat cumbersome. The confirming dialog box that appears after the selection of each function was seen by some to be unnecessary:

> *When you were adding something there was [sic] two steps. First there was the first thing and you had to click confirm or something like that. I can't remember what the second step was. If they had – if it was just one step to add something or delete something instead of ... I might have been more willing to personalize it. [Interview, User 13]*

> *The double menu that you get…I found confusing, a little bit, but it was easy to use. ... It's a little clunky. [Interview, User 17]*

> *I mean it wasn't difficult. It was just you had to click, and you had to click again, and click again, and go back and go forth – it was just bulky. [Interview, User 22]*

One participant just found that it looked "worrisome":

> *The mechanism for adding and removing features worked well, but looked "worrisome." It looked as if something was going to go wrong any second… This fear might have been caused by my many previous misadventures with Word, and by horror stories heard from friends and colleagues. [Q8, User 4]*

Our design goal for this mechanism was to make it straightforward/understandable so we opted for a design that offered only basic functionality (adding and deleting functions) and that could be learned quickly through trial and error. Despite the comments above, all participants commented on how easy it was to use the addition and deletion mechanism. Thus we believe that it was easy to use in the sense that it was easy to figure out what to do

[38] MSWord XP is the successor version to 2000.

and no errors occurred, but there were too many steps required. One participant pinpointed our tradeoff: “The Add/Delete procedure seems slow and redundant for some reason, but is rather idiot-proof” [Q3, User 24]. This could be rectified by removing the confirmation dialog box and designing a new form of menu such that when the user is selecting items from a menu to add to the personal interface, the menu stays open and check boxes appear adjacent to each item indicating its availability in the personal interface. Currently, in order to add a menu item the user selects the item as in normal menu usage; after selection, the menu disappears and the user must reopen the menu in order to add another item.

Four participants suggested having some automated assistance in the construction of their personal interfaces but some of them did acknowledge the loss of control as a potential negative side effect. Interestingly, the first quote below comes from User 7 who is annoyed by the adaptive menus and yet he still expects that the personal interface could be built automatically.

> *So if I could have something where it automatically creates the personal based on my use without having to point and click buttons around, which is easy enough but a bit of a pain, if it automatically could do it for me so that over time I created a personal interface by default so my usage pattern creates it without that annoying short menu stuff, that would be nice. Because then I wouldn’t actually have to think about it, I’d just use Word and it would create it as I go. [Interview, User 7]*

> *If it said something like “do you want to add this to your personal menu?” – if there was something like that whenever I was using the full menu, then I probably would have said “yes”. And then I would have customized it. You know sometimes they have a little popup – that probably would do it. Or else it might have really pissed me off too [laughs]. [Interview, User 13]*

> *Another idea is to have you use the program for a few times and let the program decide what you use and the program put up the personal for you. And then you can alter it as you desire. So it is sort of seamless for the user. You could use MSWord for you know, a day, a week, an hour, whatever it is and then have it analyze the data and decide what should [be in your personal interface]. But then you have the option to modify it. [Interview, User 22]*

> *It would be cool to actually track how much I was using like to be able to see the log file and go like I haven't used that in a month, I should get rid of it. You know just like cull your interface every so often you know based on what your actual usage was. I don't think that the general population would be into that… As long as it didn't go too far like you know the paperclip and started doing too much I wouldn't want a bunch of dialog boxes popping up "you haven't used this" – like out of my face. I want to control that but it would be cool to have an option to do it as long as didn't do too much for me and try to get too smart. [Interview, User 11]*

Two participants felt that MSWord Personal was "a good start" but in addition to simply selecting a subset of functions for their personal interfaces, they wanted to be able to restructure the menu hierarchies:

> *I would like to be able to rewrite the stupid menu structure of the MS Word program, not just select the options that I want within the stupid tree structure. [Q4, User 7]*

> *I would have liked to put things in different places you know. And this is bad because I do this in Word because I don't like what they have decided is on this menu. And I want to like to take part of this menu and attach it and put it with this menu and move it around and stuff. [Interview, User 11]*

Both of these participants were surprised when they were informed at the end of the study that this restructuring functionality is available through MSWord's native customize facility. Relative to our personalizing mechanism, the native facility is very sophisticated and requires substantially more skill to use. Introducing this form of customization into the personalizing mechanism would likely make it inaccessible to non-advanced users. (It is worth noting that these comments were made by Users 7 and 11, both of whom are expert long-term users of MSWord.)

Three participants wanted to be able to add an entire menu at once. The current design requires each menu item to be added one at a time, and because of the number of steps involved this can be time consuming if the majority of a menu is desired. This problem would be resolved using the new form of menu with checkboxes as previously described.

> *It would be nice if there was some way to add multiple functions to my personal interface all at once. For example, if I wanted to add all the functions in the "Tables" drop down menu, it would be nice if I could do it all at once instead of one at a time. [Q5, User 10]*

> *Adding entire menu sections would be helpful. For example, adding the Table menu option required manually adding all of its sub-options. A process that required about four or five "adds" when it would have been nice to have an option to simply select the entire menu option category. [Q4, User 7]*

> *I realized that I use menu items more than buttons. So one of the things I would have liked to have the option to have the entire menu…so that I didn't have to go back over to the full interface to see what else I was forgetting. [Interview, User 21]*

Two participants felt that the personal interface should have initially included more functions – those functions that are used by everyone. The implicit assumption is that such a set exists.

> *If there was an interface, maybe it differs by industry, I don't know, but with the most common functions – like print. Everyone prints. Everyone makes things bold. So if there was a kind of pre-selected simple menu that wasn't overwhelming. I would maybe be tempted to prefer something like that. Like it would make my decision harder to decide should I use the full 2000 interface or the personal interface and then be able to switch. That distinction would be a little less clear in my mind. So if there was a pre-selected bunch of functions that everyone happens to use and I happen to fit into that everyone category… [Interview, User 3]*

> *The starting interface (before I added my own selections) has too few buttons and menus. It takes a while to add all I need. [Q8, User 5]*

One participant requested the ability to have more than two interfaces – she wanted different personal interfaces related to the different tasks she performed:

> *One thing that would have been cool is if I could have had different settings. Like if you have the default Word and then a personal… because I work on charts and I work on just regular reports. If I could have two kind of different settings – say this is going to be my flowchart settings. And those would have draw, all the draw and shape tools and everything. And then this is my report interface and that would just have regular stuff. That would have been cool. And I don't know if a lot of people work like that. Because sometimes …I'm just writing*

> *reports or writing proposals and stuff like that and then other times I'm just doing very different things and I need to switch my orientation around and like have all this drawing stuff. And that just doesn't fit with regular writing. [Interview, User 11]*[39]

Finally, one participant wanted to be able to personalize the context menus that are displayed when the right mouse button is clicked[40]:

> *Those are things that I like on the right-click menu and I'd like to have been able to modify that as well. [Interview, User 17]*

Summary of Multiple-Interfaces Design Experience

The majority of participants had a positive experience of MSWord Personal. Fourteen out of the 19 participants (74%) for whom we have all logging data spent 50% or more of their time in their personal interface and added all of their frequently used functions (those used on 50% or more of the days that word processing occurred). In general, participants added functions according to their usage in that the most frequently used functions were the ones most likely to be added.

The mean ratings for the ease of use, the flexibility, and the intuitiveness of the personalizing mechanism were all high, however, the qualitative feedback suggests that a number of design improvements are needed. The main complaint was that the confirmation dialog boxes caused the mechanism to be somewhat clunky. This design flaw would be easy to rectify.

There wasn't a dominant approach to personalizing and using the two interfaces. The feature-keen participants were approximately twice as likely to add all used functions to their personal interface rather than just those that are frequently used, whereas the feature-shy were more evenly divided. On the other hand, the feature-shy were more than three times as likely to add features on an as-needed basis rather than simply adding all the functions that they would ever use up front. The feature-keen were more evenly divided on this aspect of

[39] We have considered other "bases for personalization" in addition to reduced functionality sets. These include task-based personalization and the use of digital personas. See McGrenere and Moore [2000].

[40] On Macintosh computers context menus are displayed by holding down the control key on the keyboard and pressing the mouse button at the same time.

the approach. Seven participants gave up to some extent on their preferred approach which, for the most part, meant that they stopped personalizing but they did not necessarily stop using their personal interface. One of these 7 participants gave up right away – he could not immediately see the value of a personal interface and it was not worth it to him to try it out.

If we look just at those participants who did not give up we see that collectively 81% of all the functions that they would eventually add were added within their first two days of using MSWord Personal. Participants were not constantly personalizing – on average, the personalizing mechanism was used on only 3.8 days, and participants had added 90% of their functions by 31% of the way through the 4 weeks. This data shows that regardless of what approach was taken, the majority of participants added the majority of their functions early in their use of MSWord Personal. This is strongly supported by an initial trigger to add functions within the first two days of usage, which accounted for almost 77% of the functions that were added. Another 9% of the added functions can be accounted for by immediately-needed functions that were added directly before they were used.

Although it is stating the obvious, it should be noted that the participants' behaviour with respect to personalizing is at least somewhat related to the design of the personalizing mechanism. Had the mechanism been designed such that functions could be added more efficiently, it may have made some participants less likely to give up. There is clearly a cost/benefit ratio at play. For some users, even if the cost to personalizing is relatively high, there may be sufficient benefit derived from a personal interface to make the cost worthwhile. For others, the benefit does not outweigh the cost. The difficulty is that the perceived cost and benefit are both dependent on individual users. For those users who perceive little benefit to a personal interface, the cost of using the mechanism has to be minimal. We expect that had our mechanism been more efficient we would likely have had a few less participants give up, but still not everyone would have stayed the course.

The expected differences between the feature-shy and the feature-keen participants did not play out in any substantial way in how they interacted with MSWord Personal and what they had to say about their experience using it. Significant differences between the two groups of participants did appear, however, in terms of how MSWord Personal was compared to other interfaces as will be shown in the next section.

5.7.2 Comparison with the Adaptive Interface

We turn now to the remaining hypotheses, which cover our second main evaluation goal, namely to compare the multiple-interfaces design of MSWord Personal to the adaptive interface of MSWord 2000.

The first four of these hypotheses (H7 – H10) compare the two interfaces with respect to satisfaction, navigation, control, and learning. In order to assess the four dependent measures a series of three factorial ANOVAs[41] was run:

1) Q1 vs. Q6: compares measures after extended time in each condition. Q1 responses reflect usage of 1 month or more with MSWord 2000. Q6 reflects 1 month's use of MSWord Personal.

2) Q6 vs. Q7: compares measures as an initial reaction of returning to MSWord 2000 after 1 month's use of MSWord Personal.

3) Q2, Q3, Q4, Q5, Q6: compares measures at regular intervals during 4-week usage of MSWord Personal.

Recall that copies of Q1 and Q2 can be found in Appendix N. (Q3, Q4, Q5, Q6 contain identical questions to Q2 are therefore not included in the Appendix.)

In addition to reporting statistical significance, we report effect size, eta-squared (η^2), which is a measure of the magnitude of the effect of a difference that is independent of sample size. Landauer [1997] notes that effect size is often more appropriate than statistical significance in applied research in Human-Computer Interaction. The metric for interpreting eta-squared is: .01 is a small effect, .06 is medium, and .14 is large.

[41] The Analysis of Variance (ANOVA) is a statistical test that compares the amount of variance between groups to the variance found within groups.

	Independent Variables		
Q1 vs. Q6	**Personality (P)**	**Version (V)**	**V X P**
Satisfy	1.12	1.27	4.12 **
Navigate	.03	5.76 ***	.05
Control	6.21 ***	4.38 **	4.38 **
Learn	4.07 **	4.13 **	2.64
Q6 vs. Q7			
Satisfy	.18	.85	.85
Navigate	.07	8.02 ***	.16
Control	.70	5.89 ***	.44
Learn	1.33	3.08 *	1.11
Q2, Q3, Q4, Q5, Q6			
Satisfy	.28	.27	.27
Navigate	.00	2.38 *	.41
Control	2.32	2.02	.64
Learn	1.90	1.56	1.10

*p<.10 **p<.06 ***p<.05

Table 5-13: F values for three ANOVAs for each of the four dependent measures. All F values have degrees of freedom F(1,18) except for those reported in the shaded region of the Q2 – Q6 analysis which are F(4,72) (N=20).

Table 5-13 shows the results of each of the three different analyses for each of the four dependent measures. Figure 5-2 shows the graphs of Q1- Q7 for each of the measures.

H7 Satisfaction Hypothesis

The MSWord versions did have a different impact on the satisfaction of the two groups of participants (Figure 5-2). There was a borderline significant cross-over interaction for Q1 vs. Q6 (F(1,18) = 4.12, p<.06, $\eta^2 = .19$) prompting us to test the simple effects for each group of participants independently. The Q1 vs. Q6 comparison was not significant for the feature-keen participants, however, the increase in satisfaction was borderline significant for the feature-shy (F(1,9) = 3.65, p<.10, $\eta^2 = .29$). Two further tests compared the satisfaction of the feature-shy participants to the feature-keen participants at Q1 and then at Q6. The feature-shy were found to be (borderline) significantly less satisfied than the feature-keen while using MSWord 2000 at Q1 (t(18) = –2.04, p<.06). However, there was no significant difference detected between the two groups while using MSWord Personal at Q6.

Interestingly, when participants returned to MSWord 2000 (Q6 to Q7) the feature-shy appear to have dropped in satisfaction and the feature-keen had effectively no change, but this cross-over interaction was not significant.

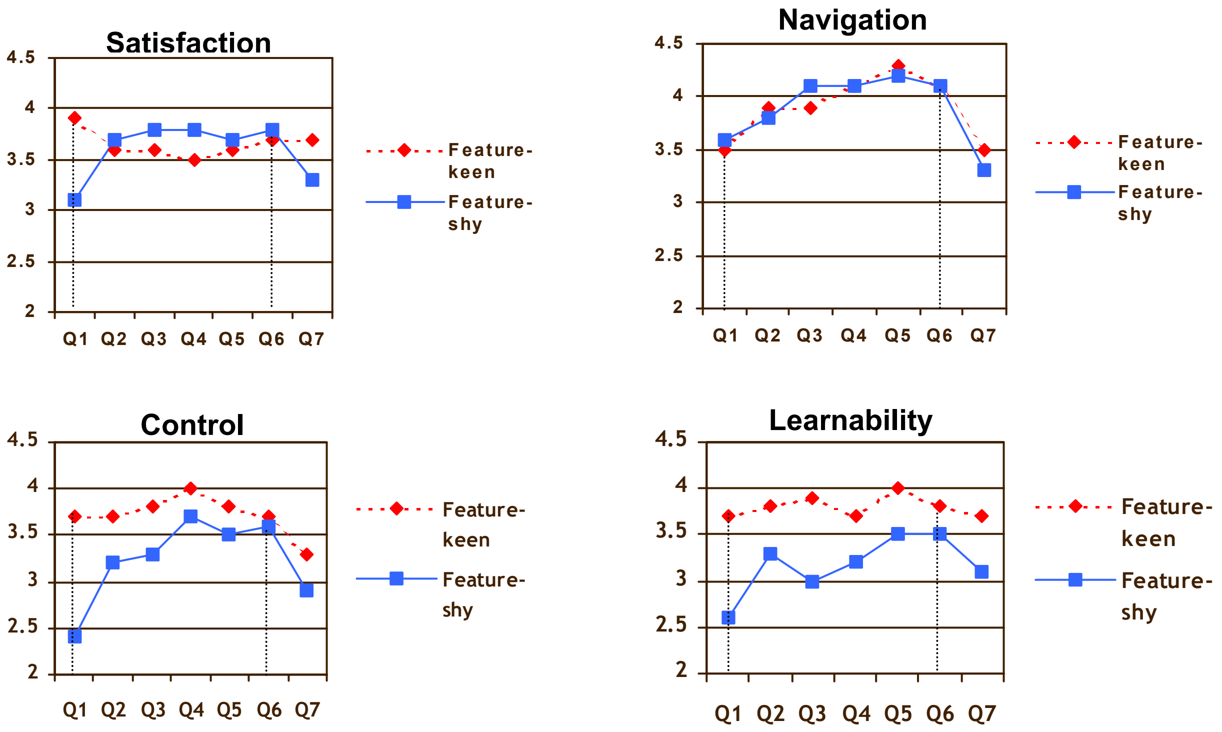


Figure 5-2: Satisfaction, navigation, control, and learnability graphs for Q1 – Q7 (N=20).

These findings suggest that the feature-shy participants were less satisfied than the feature-keen participants when using MSWord 2000, however, the feature-shy participants experienced an increase in satisfaction while using MSWord Personal. The feature-keen participants did not experience a positive or negative change in satisfaction when they switched to MSWord Personal.

H8 Navigation Hypothesis

The version of MSWord had a significant main effect on users' perceived ability to navigate in both the Q1 vs. Q6 comparison (F(1,18) = 5.76, p<.05, η^2 = .24) and the Q6 vs. Q7 comparison (F(1,18) = 8.02, p<.05, η^2 = .31) (Figure 5-2). Both comparisons favoured MSWord Personal. There was also a borderline significant learning effect in Q2 through Q6 (F(4,72) = 2.38, p<.10, η^2 = .12) indicating that navigation became easier over time, however, none of the posthoc pairwise comparisons with the Bonferonni error correction were significant.

These findings suggest that both the feature-keen and the feature-shy participants found it easier to navigate the menus and the toolbars using MSWord Personal than MSWord 2000.

H9 Control Hypothesis

The results of the Q1 vs. Q6 comparison of control are dominated by a borderline significant interaction (F(1,18) = 4.38, p<.06, η^2 = .20) (Figure 5-2). Testing the simple effects found the Q1 vs. Q6 comparison to be non-significant for the feature-keen participants, however, the feature-shy perceived a significant increase in control (F(1,9) = 11.17, p<.01, η^2 = .55). Two further tests compared control for the feature-shy participants to the feature-keen participants at Q1 and then at Q6. The feature-shy reported significantly less control than the feature-keen while using MSWord 2000 at Q1 (t(18) = −2.57, p<.05). However, there was no significant difference detected between the two groups while using MSWord Personal at Q6.

There was a main effect for control from Q6 to Q7 (F(1,18) = 5.89, p<.05, η^2 = .25) indicating that both groups of participants felt a loss of control when returning to MSWord 2000. Note that the statement being rated reflects a user's general sense of control over the software and not simply their control of the menus and toolbars.

These findings suggest that at the outset the feature-shy participants felt that they were less in control of the MSWord 2000 software than did the feature-keen participants, however, the feature-shy participants experienced an increase in control with MSWord Personal. The feature-keen participants did not experience a positive or negative change in control when they switched to MSWord Personal. Both groups of participants experienced a loss of control when they switched back to MSWord 2000 after having used MSWord Personal for 4 weeks.

H10 Learnability Hypothesis

In the Q1 vs. Q6 comparison the MSWord version had a borderline significant main effect on learnability (F(1,18) = 4.13, p<.06, η^2 = .19) showing that both groups of users' perceived ability to learn the available functions was greater with MSWord Personal than with MSWord 2000 (Figure 5-2). Personality type also had a borderline significant main effect on learnability (F(1,18) = 4.07, p<.06, η^2 = .18) showing that, independent of software version, feature-keen participants felt better able to learn the functionality offered than did the feature-shy participants.

The Q6 vs. Q7 comparison showed that the software version had a borderline significant main effect (F(1,18) = 3.08, p<.10, η^2 = .15) whereby participants' perceived ability to learn decreased when they returned to MSWord 2000.

These findings suggest that the feature-keen participants generally find it easier to learn functions than do the feature-shy participants. Both groups of participants found it easier to learn functions with MSWord Personal than with MSWord 2000.

H11 Three-way Comparison Hypothesis

In the final debriefing interview participants were asked if they could explain how the "changing menus" worked. Although all participants were aware of the short and long menus and could explain how to expand the menus, 7 of the 20 participants had to be informed that the short menus were in fact adapting to their personal usage. Participants were then asked to rank according to preference MSWord Personal, MSWord 2000 with adaptive menus, and MSWord 2000 without adaptive menus (the standard "all-in-one" style interface).

Figure 5-3 shows the frequency of the three-way rankings that participants gave for the interfaces. Of the six possible rankings, only five occurred. The feature-shy and feature-keen are shown separately, because there were differences in their rankings.

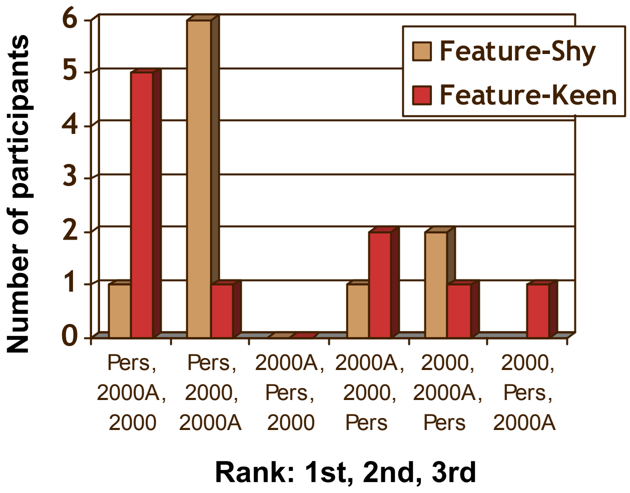


Figure 5-3: Ranking three different interfaces for MSWord: Personal, 2000, and 2000 with adaptive menus (2000A) (N=20).

We can aggregate across the rankings to make two-way comparisons. For example, by looking at the two leftmost ranking orders in the figure we see that 7 feature-shy participants preferred MSWord Personal to the other two designs. From the remaining ranking orders we see that 3 feature-shy participants ranked the all-in-one design before the MSWord Personal design. This shows that for the feature-shy there was preference for the MSWord Personal to the all-in-one design: 7 participants to 3 participants. One can repeat the same steps to find that the feature-shy preferred the all-in-one to the adaptive design (8 to 2). However, the feature-keen did not prefer the all-in-one to both the adaptive and MSWord Personal designs as expected. In fact MSWord Personal was preferred to adaptive (7 to 3) and preferred to the all-in-one (6 to 4) but the adaptive was preferred to the all-in-one (7 to 3).

We see that 13 participants in total preferred Personal to either forms of 2000. Interestingly only 2 of the feature-shy ranked adaptive before all-in-one as compared to 7 of the feature-keen. This can perhaps be explained in part by the fact that 6 of the 7 participants who were unaware of the adapting short menus were feature-shy participants. This is an indicator that lack of knowledge that adaptation is taking place may contribute to overall dissatisfaction with an adaptive interface.

The difference in opinion about the adaptive menus came through clearly in the interview data and is covered below.

Qualitative Feedback

As before, we review the participants' comments to provide more context for the quantitative results. In particular we highlight what was said about the adaptive menus, about the challenge of working with an unfamiliar interface, and finally, thoughts on how the different interfaces impact learning.

Adaptive Menus

The adaptive menus of MSWord 2000 were liked by some and strongly disliked by many, but others had little opinion either way.

There were 3 participants who ranked the adaptive menus in Word 2000 first (H11). Two had very positive comments when asked if they were aware of these menus and if they knew how they worked:

> *Yes [I have noticed the "changing" menus], love that. It does it on my operating system as well… Yes [I know how they work], it seems that the functions that you use most often are the ones that show up. Or I don't know if they are the ones that I use most often or the ones that are used most often. I haven't figured that one out yet. …. Actually, I don't think it is the ones that I use most often. I think that it is a standard small set and then you click on the bottom and the whole set comes up. Also I've noticed if you sit there and wait and look at it will expand as well. [Interview, User 22]*

> *It seems like it just responds to whatever functions you use most recently. It gives you the most recent five or whatever. I like that kind of personalization because it is more dynamic and it just seems that I am always changing what I am doing from day to day. [Interview, User 10]*

Interestingly, both participants are expert long-term users of MSWord and although they are aware of the adaptive menus, neither of them can fully explain how they work. User 22 suspects that the menus are adapting to her usage but then questions whether this is right. User 10 knows that they are adapting but implies that there is a maximum number of items that can be shown when in fact if one has used all of the menu items recently, they will all appear in the short menu.

There were 7 participants who had very negative experiences of the adapting menus. The first comment below refers to the ordering of items when the menu expands from the short version to the long one. If, for example, menu items A, E, F, and H are shown in the short menu, the long menu will then expand to A, B, C, D, E, F, G, and H. If one scans the short menu and cannot find the desired item, then when the long menu appears one will likely have to rescan the full long menu because the newly-visible items are interspersed with those that have already been scanned.

> *I don't know why [I dislike the adaptive menus], because I know that…well I have some idea why but… Well okay, first of all, part of the advantage of having these mega menus is that*

you can hunt through them and I realize that the stuff you use more tends to show up at the top. But when you click the open, the adaptive menus, the menu shows up in the way it is normally. So if there's – if you use two functions and they are right side by side and then there's actually a bunch of stuff in between, that will show up in between. And so I find it's like I have to start all over again looking through the menus for the functionality, which I find really annoying. And I don't know why. It's confusing, I just find it more confusing. I think that's ultimately it. And I don't think of myself as a naïve user and I don't know why it bothers me so much it just does. [Interview, User 3]

… the adaptive menus are hell, I don't like them at all. So like that's a definite No – like that's almost a zero choice. I would never pick that, like I just hate it. [Interview, User 7]

I hate the menus where only your most recently used items show up first!!! [Q7, User 11]

I don't like the way that menus are displayed. [Q8, User 5]

Yah, and I didn't like it. Because it seemed that always I was just waiting around for the thing I really wanted and that was just so irritating. …Yah I want the whole – when I go for it, I want it there. And the shorter ones – I don't know what the advantage of the shorter ones is? It just seemed like every time what I wanted wasn't there. Probably that wasn't the case but you know… I was waiting for it to expand or I didn't realize that it was going to expand and I'd think "where the hell is it" and I'd go to something else and "oh, it's on there." [Interview, User 13]

Yah, I don't like that, it annoys me. Because sometimes there are some options that I don't know why they are not there and it took me, like well now I've figured it out, but when I started using it, I didn't understand why some options were popping up which weren't often options I needed. I don't like that at all. Yah, it's very annoying. …I don't get why some don't appear frankly. [Interview, User15]

Well the first thing is with the Word 2000 – I really really really dislike the – I mentioned this with my questionnaire before – the frequency of use menu. I was often making mistakes and because they only give you, I don't know, a fixed number, maybe six menu items, I tend to use a lot of different functions regularly all of the time. So I was always, you know, using that little piece [down arrow icon], and I was always making a mistake going – where is it? Where is it? Where is it? It's gone! It fell off. I found that just…I still find that incredibly

> *frustrating. So I would rather not do that and Word Personal didn't do that. So I much prefer it. [Interview, User 16]*

Four additional participants felt negatively about the adaptive menus but not to the same extreme as the previous 7 participants:

> *I don't really feel one way or another about that. In fact I'd rather it didn't do that because sometimes I forget like I'm looking for something and I'm like – oh, I can't find it, where is it? And I can't find it because it's a hidden thingy. [Interview, User 6]*

> *I don't know. I just think that it is too much. Like just show me the whole thing. I just prefer to see the whole thing. I don't know maybe because that's the way it used to be – like the whole menu was there… I know it should be better. Like I know that I should probably think that – oh yah there is less there and that is helping. But, I guess what I don't like about it is the uncertainty. Like I know that there is other stuff there and I just want to make sure that the thing I'm selecting is the thing that I need to select. And I need to see all of the options in order to be sure. And so even if I don't select something in the extended menu, I hover just to see the full menu, just to make sure. I just don't like the hiding. [Interview, User 12]*

> *Yah, I think so but I am more accustomed in the older to seeing all of the menus than the arrows and having to drag down to see what else is there. I find that frustrating. I would rather just see everything…. I always find that I can never find what I want with the menu. [Interview, User 14]*

> *I actually don't like the short menus. I'd rather have a long menu that I could order myself. [Interview, User 21]*

One participant did appreciate having only some options visible through the shortened menus but ultimately found that MSWord Personal provided a better balance for him:

> *This feeling that you will forget that certain functions are there if you leave [the adaptive menus turned] on but also the menus are way too long if you leave everything on [i.e., adaptive menus turned off so that you have the full menu]. So it's a balance between the two. That's why the Personal gave me the balance I wanted. [Interview, User 17]*

To summarize, 13 participants expressed opinions about the short/long adaptive menus in MSWord 2000. For 2 participants these menus worked very well. They were very strongly disliked by 7, and 4 were mildly negative.

Familiarity Breeds Comfort

Three participants addressed the challenge of switching to a considerably different interface from one with which they were accustomed.

> *It's hard to say because you are also trained in one beforehand so I had the full Microsoft Word interface and that's what I'd been using in various forms for years. So this new interface always seems a little bit, like you are always attached to the one you know. But if you had started out with the package and learned on it with the personal interface then you'd customize off the bat, almost everyone would probably agree that that is the best way to go. [Interview, User 7]*

> *And so it [MSWord] has the benefit of this many years to just dull your frustrations. So you don't think about things – so I've stopped thinking about what possible things could be improved because that's just frustrating. So, yah, I guess that's something that is really powerful in terms of habituation, right? And so even if a new interface is better it's still different – it's still different from what I know so there's that learning and there's that getting comfortable with it again. And so I think probably if the study was longer and I had used the personal one a lot more or yah a lot more frequently during that time I would have gotten more comfortable and more familiar and going back to Word 2000 would have been kind of like – oh this is different. But when I was back to Word 2000 it was like – oh yah this is, I remember this. [Interview, User 12]*

> ***What is the main reason that you chose Word 2000?*** *Probably because it is the one that I am most familiar with.* ***If you had learned MSWord using an interface like Personal, would your reaction be different?*** *Definitely. 100%. For sure…I've been using Word since whatever version – Word 4, Word 3 – something like that. So, I see every upgrade that I happen to see and I am used to it. But if I had had it differently from the beginning I am sure that I would find it [Personal] more to my liking. [Interview, User 22]*

For these 3 participants at least, and perhaps for others as well, it is clear that the personal interface in MSWord Personal is something very unfamiliar, much more so than the adaptive interface in MSWord 2000.

Learning

During the debriefing interview after participants were asked to rank the three interfaces (Personal, 2000, and 2000 without adaptive menus), they were then asked which of the three interfaces best enables the learning of new features. Fifteen participants indicated that MSWord 2000 without the adaptive menus was best and the standard reason given was that seeing all of the menu items all of the time gives one a sense of what is available and thereby promotes learning the available functions. Some participants specifically mentioned learning through exploration. Two representative comments are:

> *2000 without the changing menus [is best] just because you can see all of your options so you know what all of the features are. [Interview, User 2]*

> *I want to learn them all or nothing...In general I think that if they are there you are more likely to say – what is this? – and use it. So maybe a little bit. Because I have explored. The only way I've learned to use the program is by playing with it. So I saw the indexes and I went – how do you do that? – so now I know how to make an index. I guess if I never saw it, I'd probably never have played with it. [Interview, User 22]*

Two participants indicated that having everything available in the full interface within MSWord Personal supported learning equally as well as the all-in-one interface (i.e., MSWord 2000 without adaptive menus). They did not need to be accessing the full menus all of the time:

> *I think if I can switch to the full interface like that, it's very convenient. So I think the learning ability [in MSWord Personal] shouldn't be impacted. ... Just one click. [Interview, User 5]*

> *I want it all or I want mine [personal interface]. In the same way, I don't want the computer deciding what it's going to show me. I want to decide myself. If I don't know how to do something then I want to go and use the full interface more as like a reference or something and have it all kind of there. [Interview, User 11]*

Two participants indicated that they learn through exploration but that they are not in exploratory mode all of the time. Having a personal interface forces them to take ownership of learning in that they actively decide when to enter exploratory mode by switching to the full interface. Perhaps the personal interface can be seen as the working interface and the full interface is for exploring:

> *I feel very neutral about that. I think I was neutral all the way through. Because even when you have all the menu items present you still don't actually know what they do. When they are disappearing [adaptive] as I say, you don't really know what is there that you are missing. And in the personal I guess that it inclined me to go looking for things to decide whether I wanted to add it into the menu. From that point of view, the personal was better because it forced me to go look. ...* ***What about learning by remembering labels you have seen in the long menus?*** *Well it's passive. Whereas having to search for it and taking ownership of the menu is active. ... But definitely, I do take ownership. You're doing just in time learning and it's learning in context because it's something you want to do and you need to do and you look for it. [Interview, User 17]*

> *Well that's a really interesting question. I think the one with the adaptive menus doesn't support it [learning] at all because it just disappears on you and you don't even know that it is there. I would say that it is probably similar between the regular Word long menu and the Personal one. Because you still have to think that you need something different and find it. Often my strategy around that is that somebody says – Oh, try this – or – there must be a way to do this – and then go to help or whatever.* ***What about learning by remembering labels you have seen in the long menus?*** *That's not been my experience myself...I must say that whether it's the Personal sort of thing or the long menus, for me at least it's – oh, I have to go exploring, I'm going to go look. Because with your sort of routine daily functions I am not using the menus, I'm not paying attention to the menus. So I'm not in explore mode. I'm not even in attentive mode, I wouldn't even notice if I've seen something related or not. So I wouldn't usually notice. [Interview, User 16]*

None of the participants thought that the adaptive menus best supported learning.

Summary of Comparison

Unlike the results from the first set of hypotheses where there wasn't any substantial difference between the feature-shy and the feature-keen participants, some significant differences did surface in the self-reported measures from Q1 through Q7. Both groups of participants favoured MSWord Personal in terms of their ability to navigate the menus and toolbars and to learn the features. In general, feature-keen participants felt they were better able to learn as compared to the feature-shy. Results for control and satisfaction were dominated by interactions, whereby feature-shy participants experienced an increase in both satisfaction and control while using MSWord Personal and the feature-keen did not experience any significant difference. One way that this can be interpreted is that MSWord Personal improved satisfaction and sense of control for the feature-shy without negatively affecting the feature-keen.

Thirteen participants (7 feature-shy and 6 feature-keen) preferred Personal to either MSWord 2000 with or without adaptive menus, however, the two groups of participants differed in their second ranking with the majority of feature-keen choosing adaptive and the majority of feature-shy choosing all-in-one. Interestingly, of the 13 participants who expressed an opinion about the adaptive menus beyond a simple ranking, only 2 were positive. Yet 3 participants ranked adaptive first and 9 ranked it second, so adaptive menus did have some supporters. The imbalance in the comments about these menus likely suggests that those who disliked the menus had a more extreme or passionate dislike as compared to those who liked the menus.

The results with respect to learnability are interesting. When participants were asked to assess the learnability of the multiple-interfaces design and the adaptive design through Q1 to Q7, the multiple-interfaces design had significantly higher ratings. In the debriefing interview, however, the all-in-one style interface was presented as an option alongside the other two interfaces. Fifteen participants ranked all-in-one first with respect to learnability. Four participants thought that the multiple-interfaces design supported learning at least as well as the all-in-one; 2 of these 4 felt that having two interfaces forces a separation between exploratory mode and work mode and for that reason it may actually be better than the all-in-one design.

We believe that it is instructive to take a step back for a moment and highlight our findings with respect to the experimental design of Study Two. One might argue that our dependent measures of satisfaction, navigation, control, and learning are at least somewhat related to our independent variable of personality type (feature-keen and feature-shy). For example, it may not be entirely surprising that the feature-shy were significantly less satisfied than the feature-keen with MSWord 2000 at the time that Q1 was administered. After all, MSWord 2000 is a feature-laden application. Thus, if all we had done was taken the dependent measures at Q1 (and no other questionnaires) and then concluded that there was a statistically significant difference between the feature-keen and the feature-shy with respect to the satisfaction of MSWord 2000 (as well as the other dependent measures), one might rightfully say "so what?". But this is not what we have done. There was another independent variable, namely the two interface conditions (MSWord 2000 and MSWord Personal) and participants were given 4 weeks to experience the new MSWord Personal design (i.e., it was a repeated measures design) and then 2 weeks back in the Word 2000 adaptive design. The data collected for the dependent variables changed over time and it was the impact of the two interface conditions on the dependent measures that is most interesting in the findings we have reported in this section.

5.7.3 Summary

In addition to the sections that summarize the participants' experience of the multiple-interfaces design (page 136) and the comparison with the adaptive interface (page 150), we also include Table 5-14 that restates each of our hypotheses and the associated results.

Hypothesis	**Results**
Experience of Multiple-Interfaces Design	
H1 Usage The majority of the participants will choose to use their personal interface – they will find the personalizing mechanism easy to use, intuitive, and flexible enough such that they will use the mechanism to include all frequently used functions and will spend the majority of their time in their personal interface. Feature-shy will spend more time in their personal interface than the feature-keen.	The personalizing mechanism had mean ratings out of 5 of 4.3, 4.1, and 4.0 for being easy of use, intuitive, and flexible. Out of 19 participants, 14 spent 50% or more of their time in their personal interface and these same participants added all their frequently used functions. But the feature-shy did not on average spend more time in their personal interface than the feature keen.

H2 Good Idea The concept of having two interfaces will be easily understood and will be considered a good idea. The feature-shy will find the idea to be more valuable than the feature-keen.	The concept of having two interfaces was easily understood (mean rating of 4.4 out of 5). The questionnaire statement assessing the “goodness” of the concept was ambiguous and so the results cannot be interpreted.
H3 Approaches to Personalization Feature-shy users will be more likely to keep only frequently used functions in their personal interface and then switch to the full interface when an infrequently used function is needed. Feature-keen users will be more likely to have either all functions that they ever use in their personal interface or to largely ignore their personal interface and go directly to the full interface every time MSWord is launched.	There wasn’t a dominant approach to personalizing and using the two interfaces. Feature-shy participants were not more likely to keep only frequently used functions in their personal interface (4 participants) compared to all used functions (5 participants). Feature-keen participants were more likely to want all functions that they ever used in the personal interface (7 participants) rather than just the ones used frequently (3 participants).
H4 Toggling The toggling mechanism will be considered an easy way to switch between the two interfaces. Because of the different approaches to using the two interfaces (H3), the amount the user toggles between interfaces will not be correlated with how valuable the user believes the idea of multiple interfaces to be.	The toggling mechanism had a mean rating of 4.5 out of 5 for its ease of use. There was no correlation between toggling behaviour and the rating of the two-interface concept, but as noted above (H2), this rating was problematic.
H5 Growth of Personal Interface Modifications to participants’ personal interfaces will be dominated by additions and there will be relatively few deletions. At the outset there will be an initial period lasting at most 10 usage days (days in which word processing occurs) when a series of regular modifications will be made. The size of the personal interfaces will eventually reach a steady state and for the majority of the participants this steady state will occur by the end of the initial period.	Apart from seven functions that were deleted by 3 participants, the size of the participants’ personal interfaces did increase monotonically. There was an initial period when modifications were made regularly that lasted on average 2.8 days, thus less than we expected. The point at which the personal interfaces did not grow/shrink by more than 10% came on average 31% of the way through the 4 weeks – this was beyond the end of the initial period for half of the participants.
H6 Modification Triggers There will be a number of different triggers that prompt participants to modify their personal interface. There will be an initial trigger to add functions because the personal interface will otherwise be almost unusable. Another trigger will be the immediate need of a function that was not initially added but that the user expects to use in the future. In addition, participants will likely add functions that they have used from the full interface on one or more occasion and that they expect to use again. Triggers for deletion will include the mistaken addition of a function and the desire to remove functions that are not being used.	There were 485 functions added in total. The initial and immediate-need triggers accounted for 77% and 9% of the functions added respectively. The previously-used and expected future-use trigger accounted for substantially fewer functions, namely, 1% and 13% respectively. There were 7 functions deleted in total. Of these, 3 qualified for the mistaken-addition trigger and none qualified for the non-use trigger. The remaining 4 functions were deleted by one participant while she was in the process of testing out the personalizing mechanism.

Comparison to Adaptive Interface	
H7 Satisfaction Feature-shy participants will be more satisfied with MSWord Personal than with MSWord 2000. The feature-shy will be more satisfied than the feature-keen with MSWord Personal.	Feature-shy participants experienced a borderline significant increase in satisfaction with MSWord Personal but there was no significant difference in satisfaction between the feature-shy and the feature-keen while using MSWord Personal.
H8 Navigation Both feature-shy and feature-keen participants will feel that they are better able to navigate the menus and toolbars with MSWord Personal than with MSWord 2000.	Both the feature-shy and the feature-keen did report that it was significantly easier to navigate in MSWord Personal than in MSWord 2000.
H9 Control Both feature-shy and feature-keen participants will feel a better sense of control with MSWord Personal than MSWord 2000.	The feature-shy perceived a significant increase in control when they used MSWord Personal, however, there was no significant difference for the feature-keen.
H10 Learnability Feature-shy participants will feel that they are better able to learn the available features with MSWord Personal than with MSWord 2000.	Both the feature-keen and the feature-shy felt that they were (borderline) significantly better able to learn the features with MSWord Personal.
H11 Three-way Comparison When asked to compare their overall preference for MSWord Personal, MSWord 2000, and MSWord 2000 with the adaptive menus turned off (standard all-in-one interface), feature-shy participants will prefer the multiple-interfaces design to an all-in-one design but will prefer all-in-one to adaptive. Feature-keen will prefer all-in-one to both the adaptive and multiple-interfaces designs.	The feature-shy did prefer the multiple-interfaces design to the all-in-one (7 to 3) and did prefer the all-in-one to the adaptive (8 to 2). However the feature-keen did not prefer the all-in-one to both the adaptive and multiple-interfaces design as expected. For the feature-keen, the multiple-interfaces was preferred to adaptive (7 to 3) and preferred to the all-in-one (6 to 4) but the adaptive was preferred to the all-in-one (7 to 3).

Table 5-14: Summary of hypotheses and results.

5.8 Execution of the Study

Although the prototype and the study protocol had been pilot tested both through the Pilot Study (Chapter 4) and the additional pilot testing done directly before Study Two, there were still a myriad of things that might have gone wrong in this study. In general the execution of the study went extremely smoothly; the technical problems experienced were minimal and the questionnaires and meetings were completed in a timely fashion. In our Pilot Study one of the participants was forced to delete the prototype; none of the participants in Study Two encountered that problem. For reasons of completeness we list the problems that we did encounter.

The technical problems were as follows:

- User 13 decided to save her current document during her first use of the personalizing mechanism. As soon as she entered the mechanism she hit Ctrl-S which is the keyboard hotkey for **Save**. Unbeknownst to her, the document was put in a state that locked editing when she entered the personalizing mechanism and so it was saved in this state. She called the researcher within an hour of the installation of the prototype and the researcher was able to diagnose the problem and fix the document right away. This is clearly a deficiency with our implementation of the prototype. It could easily by fixed by including a check for this condition, which would first unlock the editing, then do the **Save**, and then lock it again.

- The installation of the prototype for User 17 was the most challenging of all the installs. This user had other programs that interacted with MSWord 2000 through document templates and through COM technology. These other programs were largely being unused by User 17 at the time and so they were disabled or uninstalled for the 4 weeks during which MSWord Personal was being used. Within 1 week of the First Meeting this user installed an application that unbeknownst to him interacted with MSWord and effectively deleted the document template that stores the code for MSWord Personal. A backup copy of this template was stored as part of the installation process and so through email communication the user was instructed to shut down MSWord, copy the template file to the MSWord startup file folder, and restart Word. Because the state of a user's personal interface is stored in a separate flat file that remained intact even when the template was deleted, it was not necessary to reconfigure the user's personal interface.

- User 14 had his whole system wiped clean by his system administrator 3 days prior to the Second Meeting. This participant was not very technically literate and expected that MSWord Personal and the software logger would remain intact once his system was reinstalled. Luckily User 14's log files were successfully uploaded with the backup upload script the night before his system was cleaned. He indicated that he had not used MSWord in those 3 days and thus no data had been lost.

- User 7 reported that his system was demonstrating noticeable and annoying lags while he was typing that hadn't occurred prior to the installation of MSWord Personal. We expect that this was related to the logger rather than the prototype, given that the prototype code only executes when the personalizing mechanism is in use or when a user switches between the interfaces. The participant was told that it was likely the logging mechanism rather than MSWord Personal that was causing the delays and, if need be, we could disable the logger. Our contacts at Microsoft indicated that they hadn't experienced any such problems with the IV logger. In the end the user reported that he used a program to clean his Windows registry and this seemed to eliminate the problem.

- As previously mentioned there was no usage logging data collected for User 9 after the first day of installation. The key that is set in the Windows registry to indicate logging is active was intact when the user's machine was checked at the Second Meeting and the cause of the problem was never determined.

With respect to scheduling, the following irregularities were experienced:

- Three participants each forgot one of their meetings. In two cases this resulted in starting the meeting late and in one case the meeting had to be rescheduled for the next business day.

- Two participants were very late for one meeting – approximately 30 minutes late.

- One participant forgot to bring his laptop to the Second Meeting requiring it to be rescheduled for the next business day.

- Of all the questionnaires, nine were completed 1 day late. There were only 5 participants who had one or more late questionnaires.

The technical and scheduling irregularities experienced were all seen to be relatively minor and therefore did not force any participant to be disqualified.

In addition, the researcher forgot to send questionnaire reminders on two occasions (once for each of User 13 and User 22) and forgot to update one participant's personal web page on one occasion to reflect that a questionnaire had been received (User 24). For the questionnaire reminders, User 13 knew she had a questionnaire due because she had received the reminder about a meeting and she knew that there was always a questionnaire due right before a meeting. She sent the researcher an email to request the questionnaire reminder because it contained the URL to her personal web page which had a link to the questionnaire. This user relied fully on the reminders and never even bothered to bookmark her web page. In the case of User 22, the researcher realized that the reminder had not been sent when the questionnaire was not received on the due date, and she simply sent a reminder the following day. User 24 completed Q8 twice, two days in a row, when her web page did not get updated to reflect the first submission of Q8. These three incidents provide a strong indicator of the extent to which participants relied on the reminders and their personal web pages.

When queried during the final interview about their experience participating in this study, participants used the following adjectives: easy, interesting, good, pleasant, fun, fine, enjoyable, simple, fascinating, comfortable, okay, neutral, and cool. When asked if there was anything the researcher could have done to make the experience more enjoyable or easier, no participant had any suggestion. A number of participants specifically mentioned that the combination of the email reminders for questionnaires with the embedded URLs and having short questionnaires was conducive to getting the questionnaires done on time. When asked whether or not they would participate in another study like this again, 7 participants responded "probably" and the remaining 13 responded with an outright "yes".

The compensation of the $100 gift certificate had a reasonable influence on people's decision to participate: 5 participants indicated that it had no influence, for 11 participants it had somewhat of an influence, and for the remaining 4 it had a strong influence. The influence of the $100 prize for the person who completed the most questionnaires on time had less of an influence: 14 participants indicated that it had no influence on their diligence, 2 said that it had somewhat of an influence, and 4 said it had a strong influence.

5.9 Validity of the Study

The results of this study are promising, however, there were inherent limitations and constraints to the study design that may have affected the results in some way. In most experimental designs (true experiments as well as quasi-experiments) the experimental variables cannot be fully controlled, which threatens the internal and external validity of the results. Internal validity addresses the issue of whether or not experimental treatments actually made a difference in the specific experimental instance. If a difference is found in the experiment, can it be attributed to factors *other* than the treatments? External validity addresses the issue of generalizability. To what extent can the results of the experiment be generalized to other settings and populations?

It is important to mention the limitations, or threats to validity, that existed in Study Two as they may inform the design of future studies:

1) **Reactive effect:** participants were fully aware of their participation in the study and some may have been able to deduce some of the higher-level study objectives. Given this awareness, their interactions with the software and their reported levels of satisfaction may have been different than they would have been if they were not participating in a study. Note that a reactive effect is an issue in most experimental studies that is difficult, if not impossible, to fully eliminate. This effect would have likely been more pronounced with a laboratory experiment. With a field study the tasks and the setting were natural and so this effect should have been minimized although not eliminated.

2) **Multiple-treatment interference:** participants were exposed to two versions of MSWord and there were likely effects of having used one version that were not erasable when using the second. There were in fact a few participants who specifically mentioned that familiarity with MSWord impacted their experience with MSWord Personal. Note that this problem would have also existed in a laboratory experiment in which subjects were exposed to more than one treatment, however, some interference effects can be mitigated in the lab by counter balancing the order in which the treatments are given to different subjects. In our field study we did not

control the order. One way to mitigate the interference in the field would have been to have participants use the various versions over a much longer period of time. It is likely that participants would have reached a higher level of familiarity with the multiple-interfaces design had they been exposed to it for 6 months or 1 year.

3) **Researcher interference:** a single person performed the role of the researcher for this study. While this may be preferable to having multiple researchers because it mitigates differential execution between researchers, it does leave open the possibility that there was something specific to the particular researcher that systematically affected the results.

4) **Selection bias:** participants were a self-selected group because we did not have a sampling frame of all MSWord 2000 users and could not therefore draw a simple random sample. As mentioned in Section 5.3, our sample is likely not fully representative of the target population of MSWord 2000 users.

The best way to ensure that there wasn't anything incidental in our study execution that determined the results is to replicate the study. Ideally we would want to conduct a longer field study with a different person acting in the researcher role. Counterbalancing the order in which software versions are used would be ideal. In addition, using a different application, whether it be another word processor or another general productivity application, would go a long way to ensure the generalizability of the results to the class of word processing applications or general productivity applications as a whole.

5.10 Hypotheses for Future Work

Participants ranked 13 statements in questionnaires Q1 through Q8 and yet our hypotheses only dealt with 4 of them. One reason for having so many statements was the possibility of collapsing some statements together into scale variables, in the same way this was done to create the Feature Profile Scale described in Section 3.3. However, the data collected in this study did not "hang together" sufficiently reliably to allow for scale construction. Despite this, we completed the same analysis for the remaining nine statements as was done for the four reported in Section 5.7.2. The results are shown in Table 5-15 and Figure 5-4. In order to protect against Type I error[42], these findings need to be interpreted with caution. At best they should be treated as hypotheses that will guide future research.

	Independent Variables		
Q1 vs. Q6	**Personality (P)**	**Version (V)**	**V X P**
Ease of use	.40	.72	1.99
Menu control	1.04	1.57	.29
Engagement	.36	17.31 ****	6.23 ***
Match needs	2.07	1.86	.83
Getting started	0.00	.47	.47
Flexibility	.18	3.41 *	.72
Finding options	1.28	12.90 ****	4.64 ***
Discoverability	6.93 ***	5.44 ***	5.44 ***
Quickness	1.32	.06	1.43
Q6 vs. Q7			
Ease of use	.76	2.67	6.0 ***
Menu control	1.28	8.24 ***	.67
Engagement	.27	.72	1.99
Match needs	.165	7.95 ***	.12
Getting started	.231	1.00	1.00
Flexibility	1.75	3.57 *	.09
Finding options	1.00	3.57 *	1.29
Discoverability	4.35 **	2.66	0.00
Quickness	0.00	0.00	.20
Q2, Q3, Q4, Q5, Q6			
Ease of use	.65	.39	1.41
Menu control	.01	1.88	.34
Engagement	.67	1.21	.30
Match needs	.01	.44	.71
Getting started	.16	2.39 **	.42
Flexibility	.62	1.89	.99
Finding options	0.00	1.30	.62
Discoverability	3.66 *	1.77	.91
Quickness	.01	.71	.18

*p<.10 **p<.06 ***p<.05 **** p<.01

Table 5-15: F values for three ANOVAs for each of the nine dependent measures not discussed in the main body of this chapter. All F values have degrees of freedom F(1,18) except for those reported in the shaded region of the Q2 – Q6 analysis which are F(4,72) (N=20).

[42] Type I error is a rejection of the null hypothesis when the null hypothesis is indeed true. In other words, the analysis *appears* to show that the treatment has had a significant affect on the dependent measures, when in fact it has not.

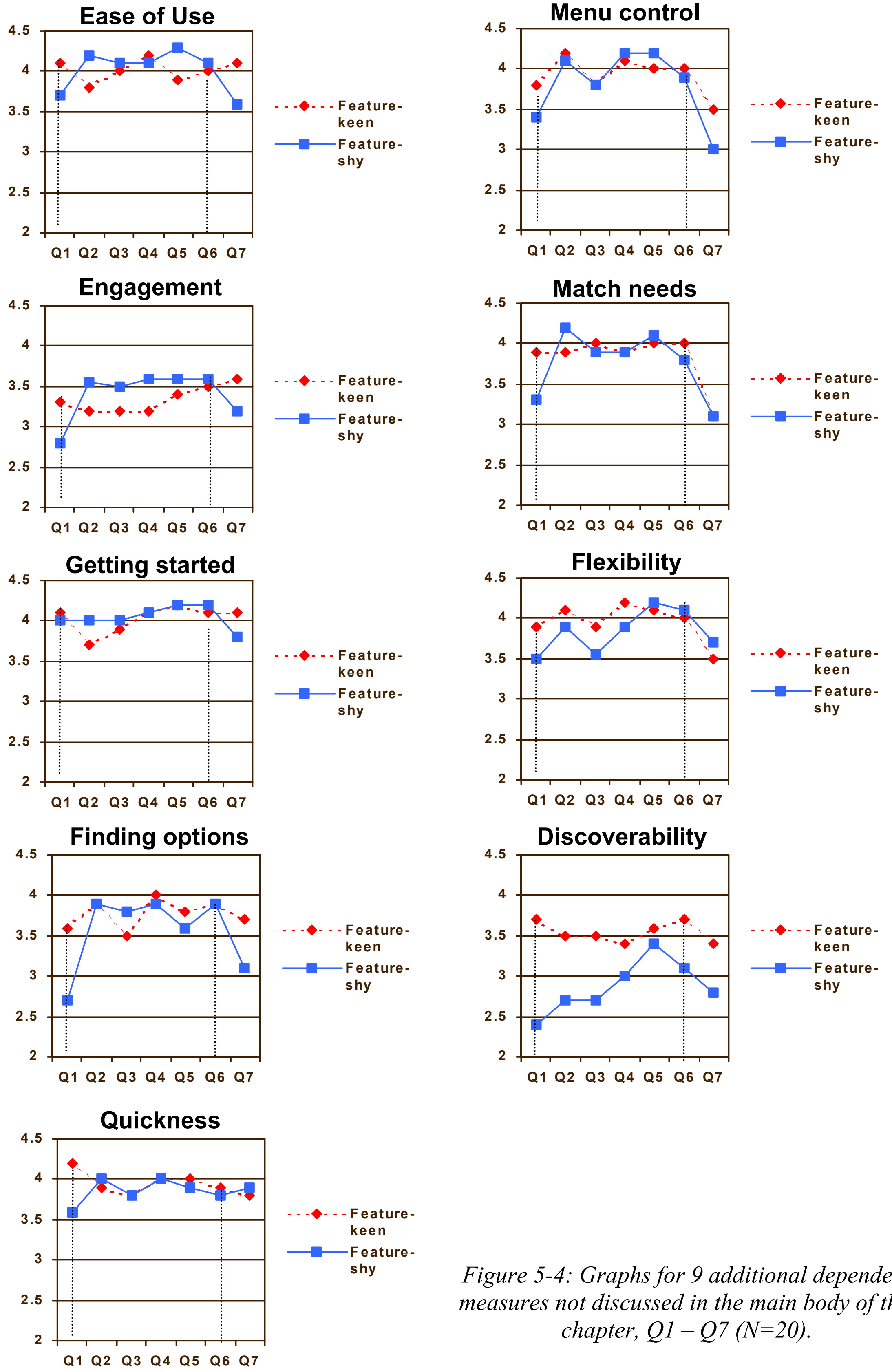


Figure 5-4: Graphs for 9 additional dependent measures not discussed in the main body of this chapter, Q1 – Q7 (N=20).

We briefly highlight some of the results that appear of interest from Table 5-14 and from Figure 5-4.

Finding Options

The ability to find options was significantly affected by the software version. The Q1 vs. Q6 comparison was dominated by a significant interaction (F(1,18) = 4.64, p<.05, η^2 = .21). Testing the simple effects found the Q1 vs. Q6 comparison to be non-significant for the feature-keen participants, however, the feature-shy perceived a significant increase in “findability” when using MSWord Personal (F(1,9) = 17.05, p<.01, η^2 = .66). Two further tests compared the ability to find options of the feature-shy participants to the feature-keen participants at Q1 and then at Q6. The feature-shy were found to be (borderline) significantly less able to find options than the feature-keen while using MSWord 2000 at Q1[43] (t(18) = −1.80, p<.10). However, there was no significant difference detected between the two groups while using MSWord Personal at Q6.

There was also a borderline significant main effect in the Q6 vs. Q7 comparison where ease of finding options decreased when participants returned to MSWord 2000 (F(1,18) = 3.57, p<.10, η^2 = .17).

Discoverability

Q1 vs. Q6 comparison was dominated by a significant interaction (F(1,18) = 5.44, p<.05, η^2 = .23). Testing the simple effects in the Q1 vs. Q6 comparison showed that the feature-shy group found it easier to discover new functions using MSWord Personal (F(1,9) = 21, p<.01, η^2 = .70) but there was no significant difference for the feature-keen. Two further tests compared the ability to discover new features of the feature-shy participants to the feature-keen participants at Q1 and then at Q6. The feature-shy were found to be significantly less

[43] Given the similarity between the questionnaire statement used to rate the ability to find options (see Section 5.5) and the statements used to construct the Feature Profile Scale (see Appendix G), which categorizes users to be feature-keen, feature-neutral, or feature-shy, it is not surprising that there was a statistically significant difference between the feature-keen and feature-shy at Q1 for this dependent measure.

able to discover new features than the feature-keen while using MSWord 2000 at Q1[44] (t(18) = −3.04, p<.01). However, there was no significant difference detected between the two groups while using MSWord Personal at Q6.

The Q6 vs. Q7 and the Q2 through Q6 comparisons both had main effects for personality whereby the feature-shy group's self-reported ability to discover new functions was lower than that of the feature-keen group (F(1,18) = 4.35, p<.06, η^2 = .20 and F(4,72) = 3.66, p<.10, η^2 = .169 respectively).

Ease of use

The Q6 vs. Q7 comparison had a significant cross-over interaction (F(1,18) = 6.0, p<.05, η^2 = .25). Testing the simple effects found the comparison to be non-significant for the feature-keen participants, however, the feature-shy perceived a significant decrease in ease of use when they return to MSWord 2000 (F(1,9) = 9.0, p<.05, η^2 = .50).

Flexibility

The Q1 vs. Q6 and the Q6 vs. Q7 comparisons both had borderline significant main effects for software version that favoured MSWord Personal (F(1,18) = 3.41, p<.10, η^2 = .17) and F(1,18) = 3.57, p<.10, η^2 = .17 respectively).

The hypotheses that might be posed based on the observations above are as follows:

H12 **Finding Options Hypothesis:** Feature-shy users will find it easier to find options in the menus and toolbars with MSWord Personal than with MSWord 2000.

H13 **Discovering Options Hypothesis:** Feature-shy users generally find discovering new features that they need to be more difficult than feature-keen users. In addition, feature-shy users will find it easier to discover new features in MSWord Personal than in MSWord 2000.

[44] Similar to the statement about finding options (see previous footnote), the statement for discovering new options in the questionnaires is closely related to statements used to construct the Feature Profile Scale. It is therefore not surprising that a statistically significant difference between the feature-keen and feature-shy was found at Q1 for this dependent measure.

H14 Easier to Use Hypothesis: Feature-shy users will find MSWord Personal easier to use than MSWord 2000.

H15 Flexibility Hypothesis: Feature-shy and feature-keen users will find MSWord Personal to be more flexible than MSWord 2000.

5.11 Summary

The execution of Study Two went extremely smoothly and the results certainly encourage further exploration of the design space for multiple interfaces.

Thirteen participants ranked MSWord Personal first, preferring it to both the adaptive interface of MSWord 2000 or an all-in-one style interface. Not surprisingly, this group of 13 is almost identical to the group of 13 who did not give up on their desired approach to personalization. One participant who didn't give up did not rank MSWord Personal first and one participant who did give up did rank it first. The design of the personalizing mechanism was somewhat "clunky" in that the number of mouse clicks required to add a function to the personal interface was seen by some to be too many. A more efficient mechanism may have resulted in even more participants ranking MSWord Personal first.

One of the most interesting observations is that while there were no substantial differences between the feature-keen and the feature-shy participants in terms of how they used the two interfaces and how they approached the task of personalizing, there were a number of significant differences observed in terms of the self-reported measures. The feature-shy felt significantly more satisfied and a significantly greater sense of control with MSWord Personal than with MSWord 2000 whereas there were no significant differences detected for the feature-keen on these measures.

CHAPTER 6 Software Logging

One of our research goals has been to evaluate users' experiences with our prototype software. In order to fulfill this goal it has been necessary to use multiple data collection methods while users interact with the prototype – methods that capture different aspects of the users' experiences. Software logging is one of these methods. A software logger is a program that captures the user's interactions with a software application (or applications) in a time-based sequence and thereby provides a very quantitative view of the users' activities. It can be contrasted with more qualitative methods such as questionnaires and interviews that we have also used in our research. These latter methods ground and contextualize the data collected from software logging and collectively the methods provide a relatively broad view of users' experiences.

Our awareness of Linton et al.'s research of MSWord command usage (described in Section 2.3.2) gave us false confidence that logging technology existed for MSWord, so before launching into our Pilot Study (Chapter 4) we dedicated little thought to the specifics of logging. It was simply assumed that the logger used by Linton and his team would meet our needs; we would be able to borrow or replicate the technology. These early notions soon vanished, however, and we realized that capturing users' interactions, or logging, is not as straightforward as we had initially assumed.

This chapter is not intended to be a survey of logging technology but rather a description of our experiences with four particular software loggers that were used in various stages of our research. Hilbert and Redmiles [2000] provide a recent and comprehensive survey on the topic of extracting usability information from user interface events. They present a framework to help HCI practitioners and researchers compare and categorize approaches that have been used or may be used in the future. Hilbert and Redmiles's particular focus is on deriving meaning from voluminous data. They implicitly assume a properly constructed data stream that contains meaningful events. They note that many of the analysis techniques have never been realized beyond the research prototype stage and therefore have not been evaluated in practice. Their survey covers what should be relatively straightforward approaches, such as the transformation of an event stream through selection (perhaps of particular event types) and the calculation of counts and summary statistics, as well as more complex approaches such as sequence characterization. Our new contribution to the topic of logging is somewhat different in that it focuses on performing logging in actual practice. Although we too are concerned with log file analysis, we focus much more on issues related to capturing the log data in the first place. There are a number of pragmatic issues that need to be considered when capturing log data, which include user privacy, file format and size, and user management of the logging process.

The first logger we describe, OWL, was developed by Frank Linton's team at the Mitre Corporation. The second logger, Microsoft Tracker, was developed and used internally by Microsoft Corporation. The third logger we tried, GUI-Tester, is a commercial product developed by the company Ergolight. The fourth and final logger, Instrumented Version (IV), was also from Microsoft, again for internal use. We begin this chapter by describing some general criteria against which loggers can be evaluated. Each of the four loggers are then described and evaluated with respect to both these general criteria and the specific needs dictated by our study objectives. We conclude with some thoughts about an ideal logger and the necessary associated tools.

6.1 General Criteria

As we gained experience with the four loggers that were used in various stages of our work, we informally weighed the pros and cons of each logger with respect to our research objectives. Through this activity a set of general criteria began to emerge with respect to which any logger can be evaluated.

1) **How interactions are captured:** Are interface interactions captured and, if so, are they captured at a low level such as cursor movements, mouse clicks, and character input, or at a higher level such as menu or toolbar selection? The distinction being made here is whether an entire gesture is captured or just the final part of the gesture. For

example, a logger may record as a single event that the **Symbol** item was selected from the **Insert** menu or it may record that the **Insert** menu was clicked open, and then the user hovered up and down the menu three times with the mouse before finally selecting **Symbol** four seconds after the menu was initially opened. Often this fine-grained information is not needed, but an example where having detailed spatial and temporal data could be useful would be the case of a researcher studying menu size to determine the optimal size for error avoidance. Some loggers record only internal function calls rather than interface interactions. This can be useful if one is interested in general operations done by the user (as in task analysis) or perhaps what pieces of code are being executed, but may not be sufficient if we are interested in the user's experience with the interface.

2) **Log file format:** In what format is the data captured? Loggers may output data in simple text format, a proprietary file format for which some form of decoder is necessary, or a non-proprietary format such as a commercial database. In all formats it is possible to access the data but one format may be more convenient than another depending on how the data will be processed. For example, for aggregating the data across all users, a database format may be most convenient. Related to file format is the size of the output file and the existence, granularity, and units used for timestamps.

3) **Degree of user involvement required for use:** What, if anything, is the user required to do in order to manage data capture? Some loggers need to be started/stopped explicitly and others work virtually seamlessly, running undetected by the user. This is likely to have implications for where the logger can be used – in a laboratory or in the field. Either variant is probably sufficient for a laboratory-style study in that these studies are somewhat artificial and thus requiring the user to start and stop the logger will not likely be a hindrance. In a field-style study, however, requiring participants to manage the logger may be problematic if it adds an artificial component to what is supposed to be natural interaction. It may also significantly increase the possibility of missed data capture due to failures on the part of the participants. Some loggers, even

if launched automatically, do allow the user to temporarily disable data capture, which is related to the next issue, that of privacy.

4) **Privacy:** Is content data such as keyboard input captured, or just the events related to commands? If all keyboard input is captured, can the user easily control when capture is taking place? Capturing keyboard input has definite implications for privacy. For a laboratory-style study this is likely of little concern because such studies rarely have subjects working on their own tasks. In field-style studies, however, subjects are usually doing their own tasks, some of which are private. Subjects should be clearly told if their keyboard entry is being captured, which may result in some choosing to withdraw from a study.

5) **Processing log files:** How easy is it to extract meaningful data from the log files? Does the logger have associated tools for data analysis? If not, what steps, such as parsing, are required to process the log file? How easy is this to do? What documentation exists to assist a researcher in accomplishing all of these tasks?

6) **Robustness:** Are interactions captured robustly or are there some that do not get captured? Are there irregularities in the data capture? Is the logger stable or does it crash occasionally? Clearly, a robust application, logger or otherwise, is desirable.

There are some obvious tradeoffs between these criteria:

- Other things being equal, comprehensive data capture that includes low-level interactions (and perhaps other levels as well) will generate larger log files than capturing only high-level interactions. Because of the detail recorded, capturing low-level interactions will also require more sophisticated post processing to extract meaningful information from the log file.

- A particular file format may be desirable because of the processing it enables (e.g., a database format) but may include overhead resulting in a larger file size than plain text. A compressed file format will keep the file size as small as possible but will then require an additional processing step to uncompress the log into a usable format.

- Although having users manage the logger will likely detract them from their real tasks, it may be necessary in order to ensure privacy.

It is important to note that the extent to which a given logger satisfies any of these criteria will often be relative to the desired goals of a particular instance of software logging. In the discussion that follows, we will focus on our own logging requirements.

6.1.1 Our Logging Objectives

As noted above, how well a logger meets these general criteria is related to the research objectives for which logging is being used. Our research objectives with respect to logging were first to understand function use and frequency of function use in MSWord. We then looked at specifics about personalizing and toggling behaviour in our MSWord Personal prototype. Given that we have defined functions as "visually specified affordances" (see Section 3.1.1), our goal has been to distinguish how users interact with the software during high-level interface interactions such as menu and toolbar button selection. Our studies have been run in the field, so a small log file size has been preferable, privacy has been an issue, and mitigating the amount of work required by participants to operate the logger was desirable. The actual file format has been somewhat of a lesser concern (except for how it relates to file size, which was a concern for us). If a logger had associated tools that would have allowed us to extract the attributes of interest, then its file format would have played hardly any role in our criteria. Without tools, however, working with the text format or something that is easily convertible to plain text was desirable because it is easy to process text files.

6.2 Four Software Loggers

For each of the four software loggers investigated, we outline the technical implementation details we encountered, we assess the logger with respect to the general criteria given above, we provide a sense of the output generated by showing the actual output for the print command when it is invoked in three different ways, and we assess how well the logger met our research needs.

6.2.1 OWL

OWL (Organization-Wide Learning) was the first logger considered for the Pilot Study.

Linton et al. created the OWL logger to capture users' command usage in MSWord 6.0 in the context of everyday tasks in the workplace [1999, 2000]. Their research goal was to compare command repertories of users who perform similar tasks and thereby provide learning opportunities through recommender technology to others in the same workgroup. (See Section 2.4.4.)

Technical Details

OWL works within the macro programming available in MSWord. We first explain some basics of this facility and then describe how OWL makes use of the facility. Similar techniques were employed to implement our personalizing and toggling interface, which also relied on Visual Basic for Applications as a mechanism for modifying the way that the interface behaved.

Macro Programming in MSWord

MSWord is programmable through Visual Basic for Applications (VBA). VBA exposes Application Programming Interfaces (APIs) that can be implemented or invoked with code written in the Visual Basic programming language. This form of programming is commonly known as macro programming. There are two sets of APIs that allow the user to retarget the function call structure within MSWord (see Figure 6-1)[45].

The first set of APIs is available through the set of functions known as Word Basic functions, which include many system-defined operations such as save, cut, copy, paste, and print. If a macro is created with the same name as one of the Word Basic functions, the system will invoke it rather than the system-defined function. Within a macro the system-defined functions can be accessed by appropriate qualification syntax. Retargeting the Word Basic functions essentially enables one to change the behaviour of some of the standard operations.

[45] The two ways a function can be retargeted are strongly linked to what Hsi and Potts [2000] refer to as the morphological view and the functional view (Section 2.1).

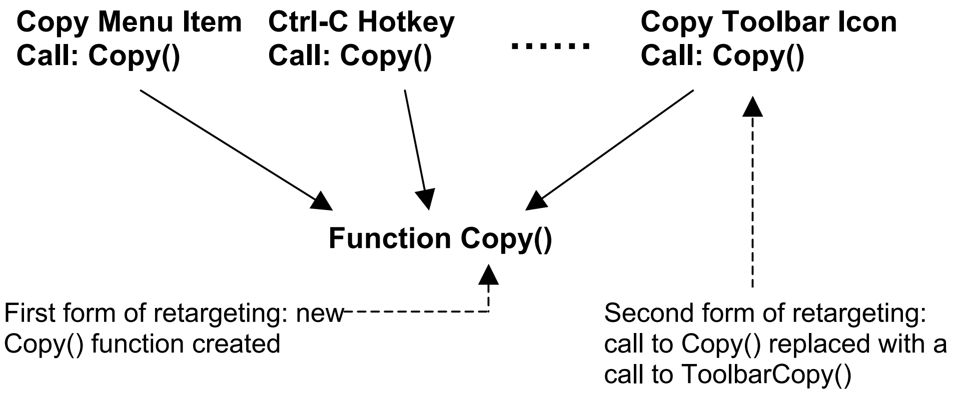

Copy Menu Item
Call: Copy()
Ctrl-C Hotkey
Call: Copy()
......
Copy Toolbar Icon
Call: Copy()
Function Copy()
First form of retargeting: new Copy() function created
Second form of retargeting: call to Copy() replaced with a call to ToolbarCopy()

Figure 6-1: Two forms of retargeting available in MSWord macro programming.

Note that there are often multiple ways for the user to invoke a given Word Basic function. For example, the Copy() function can be invoked through the copy menu item, the copy toolbar icon, by hitting Ctrl-C, or through a user-defined macro. (Ctrl-C is accomplished by hitting the "C" key while holding down the "Ctrl" modifier key. Using keyboard input as an accelerated way to invoke a function is often referred to as a hotkey, accelerator, or shortcut key.) When a new Copy() function is created all of the above invocations are affected in that they will use the newly created function rather the system-defined one.

A related but different set of APIs allows a second form of retargeting in which the user sets the function that will be called when a given interaction occurs. For example a user might want to have the copy toolbar icon call a function other than Copy(), perhaps ToolbarCopy(). This form of retargeting is isolated to the toolbar invocation of Copy() and it does not impact other calls to the system-defined Copy().

OWL works within this macro API structure by adding a whole layer of function calls to the software architecture using just the first form of retargeting described above – it is essentially a layer of wrapper functions. OWL does not use the more selective second form of retargeting. The newly created functions output an item to the log file using logging functions specific to OWL and then call the system-defined Word Basic function using a qualifying prefix to specify the Word Basic version of the function. For example, the Word Basic function for copy is EditCopy and so OWL creates its own EditCopy() function[46]:

```
Public Sub EditCopy()
LogActiveWordCommand ("EditCopy")
WordBasic.EditCopy
End Sub
```

General Criteria

How interactions are captured

OWL captures interactions at the level of the underlying function call, not at the level of interface interactions.

Log file format

OWL creates a separate log file for each document that the user edits. The file format is text. Timestamps are included with each log entry.

Degree of user involvement required for use

OWL can be turned on and off by the user through a toolbar button in MSWord. This button acts as a simple toggle and the label clearly shows "OWL is ON" or "OWL is OFF" depending on the status. The default on startup, however, is for capturing to be turned on therefore the user is not obligated to manage the logging.

Privacy

OWL does not record general keyboard input but it does record filenames when a file is opened or closed, thus raising potential privacy concerns. Users are made fully aware that OWL is active through the toolbar button mentioned above and are able to deactivate the logger at any time.

[46] In a number of cases there are irregularities with the API to Word Basic functions and so some of the wrapper functions created by OWL do require additional code.

Processing log files

Beyond the core functionality of capturing commands, OWL can also be set up to send the logs from a participant's machine directly to a specified server via a network connection. For the research conducted by Linton et al. the logs were periodically loaded from the server into a database for analysis. The capabilities of the analysis tools used by Linton et al. are unknown to us.

The log files themselves are in text format and consist only of timestamps and command names, so they are easily parsed.

Robustness

We found that OWL does not capture approximately twenty-five percent of the toolbar buttons. See the discussion below in Goodness of Fit to our Research Objectives for one possible explanation for this.

Example: Comparing Three Ways to Print

Here we compare the log output generated from invoking the print command in three different ways. All three methods result in identical output from OWL because it is the print function call that is being logged rather than the interaction with the interface.

Icon on Standard Toolbar:

```
FilePrint
```

Menu item in File Menu:

```
FilePrint
```

Hotkey Ctrl-P:

```
FilePrint
```

OWL and the other three loggers investigated all include some form of timestamps. For simplicity we have omitted these here.

Goodness of Fit to Our Research Objectives

Our goal was to distinguish how users interacted with the software at the level of interface interaction rather than simply providing the underlying function calls. Thus, based on preliminary investigation, OWL did not appear to provide a very good fit to our research objectives.

Although OWL only implemented the first form of retargeting, we experimented with using both forms of retargeting together in the hopes that we would be able to capture the underlying function calls as well as the actual interface elements invoked. We used Visual Basic for Applications to add the retargeting to the interface elements, in addition to the retargeting that OWL provided for the Word Basic functions. This worked correctly for the majority of interface elements but, unfortunately, not all. Twenty-five percent of the toolbar elements in the **Standard** and **Formatting Toolbars** could not have their function calls retargeted from the point of invocation[47]. On closer examination we discovered that OWL was not logging any of the functions called by these interface elements. In other words, OWL was not retargeting the underlying function calls and so these commands were completely missing from the log files. We were not able to find these functions in the set of Word Basic functions, which probably explains why the functions were not being logged.

OWL was considered to be unsatisfactory for our research given the large number of functions that were not being logged. The fact that we had to augment it to use both types of retargeting would have been acceptable, had all of the functions we were interested in been accessible as Word Basic functions to OWL and to us.

6.2.2 MSTracker

MSTracker was the second logger we considered for the Pilot Study and the one we ultimately chose to use for the Pilot Study. It is an internal tool used by the usability team at

[47] It was never entirely clear what differentiated the toolbar elements that could be retargeted from those that could not. All the toolbar elements are in the same object hierarchy and have methods that enable retargeting. The static analysis done automatically by the Visual Basic Editor passed successfully for all toolbar elements but errors were generated at runtime for the 25 percent of the elements that could not be retargeted.

Microsoft, and through personal contacts we were granted permission to use it for a limited period of time.

Technical Details

MSTracker is a general logger for MSWindows applications; it runs independently from MSWord. Although we are not privy to many of the implementation details of MSTracker, it generally works by listening to the Microsoft Active Accessibility (MSAA) events that are generated from interactions with most MSWindows applications, including MSWord. MSAA was not designed to facilitate software logging but rather to provide alternative displays to the standard monitor. By generating MSAA events, applications provide information about the contents of the computer screen. This information can then be used by accessibility aids such as screen readers, which enable blind users to interact with graphical user interfaces. Any standard graphical element (e.g., a standard menu or button on a toolbar) will automatically generate an MSAA event when it is selected. If an application makes use of non-standard graphical elements, however, the application itself must explicitly implement the MSAA APIs to generate an event.

General Criteria

How interactions are captured

Only low-level interface interactions are captured. MSTracker generates log output from display update events and therefore has no knowledge of the underlying application semantics such as function calls. The output is extremely detailed. It does not simply log that a menu item or toolbar item has been selected, but rather it logs all the low-level interactions that cause display updates such as mouse up, mouse down, menu popup, and hovering over a menu item. For example, simply selecting a menu item may result in 10 to 150 items in the log file. This level of detail could potentially be very useful if one was interested in low-level search issues like how long it takes a user to select a menu item once the user has entered the menu structure, but for our research this was not important.

Log file format

MSTracker outputs log files as MSAccess[48] databases. The advantage of this file format is that the log can be queried but the disadvantage is the overhead in the file size; informal testing found that even a database that contains no records is approximately 115KB in size and populated database files are generally at least twice the size of a text file with the equivalent content. This overhead may seem insignificant given the current state of cheap storage and expanding network bandwidth, however, it can have a negative impact when a lot of data is being logged and when multiple log files are being created per participant.

Degree of user involvement required for use

MSTracker must be managed explicitly through its graphical user interface. Although the application can be launched on the command line, to actually start capturing data a filename must be provided through a dialog box and the user must select the start menu item. There is an analogous stop menu item. There is no way to provide a file name and to start/stop the logger from the command line. There is also no macro facility that allows the start/stop process to be streamlined for the users.

Managing the logger can be cumbersome for users. Requiring them to manage the logger also means that they must be fully relied upon to work the logger properly. In the Pilot Study, participants unanimously complained about the need to start and stop the logger and all admitted to not starting the logger on certain occasions when they were in a hurry.

Privacy

MSTracker does record keyboard entry so there are potential privacy concerns. Hotkey functions can only be detected by parsing the text entered, although this is not as straightforward as one might assume (see below). In addition, MSTracker not only logs interactions with MSWord but also interactions with all other active MSWindows applications that implement the MSAA API. So the privacy issue extends beyond just document content to subjects’ use of the World Wide Web, email, etc.

[48] MSAccess is a database application that is included in some versions of the MSOffice Suite such as MSOffice Professional.

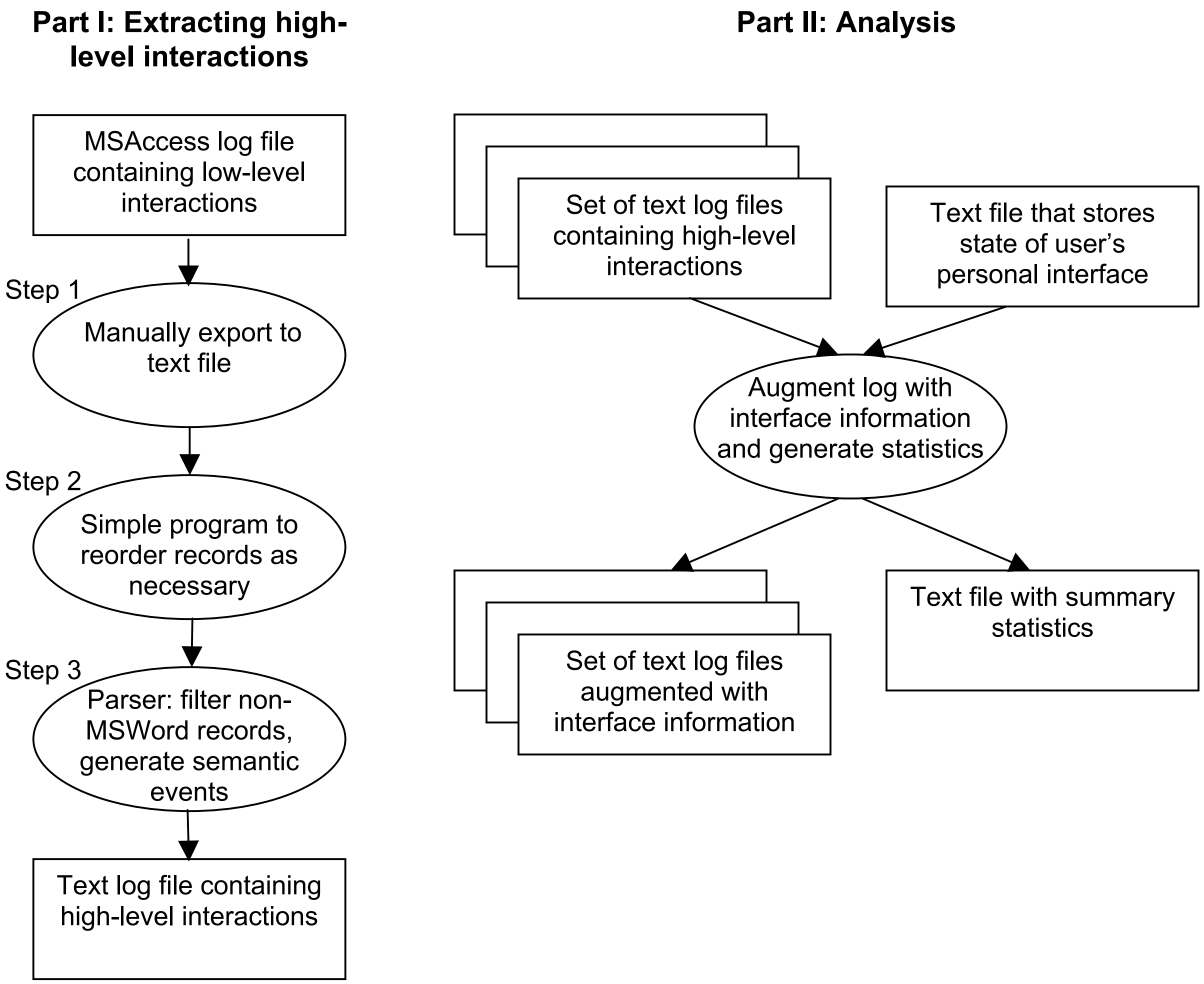

Part I: Extracting high-level interactions
Part II: Analysis
MSAccess log file containing low-level interactions
Step 1
Manually export to text file
Step 2
Simple program to reorder records as necessary
Step 3
Parser: filter non-MSWord records, generate semantic events
Text log file containing high-level interactions
Set of text log files containing high-level interactions
Text file that stores state of user's personal interface
Augment log with interface information and generate statistics
Set of text log files augmented with interface information
Text file with summary statistics

Figure 6-2: Processing MSTracker log files.

Processing log files

The output generated by MSTracker is so detailed that it must be processed in some fashion in order to provide meaningful information. There are no data processing tools provided with the logger, so we had to write our own. The series of processing steps that we used is described here and is also summarized in Figure 6-2.

In our Pilot Study participants generated one log file for each day that MSWord was used, resulting in many log files that needed to be processed. (We decided against using a single file across all days because of the potential for file corruption, which could lead to total data loss, and because the size of that one file would have been so large that it would have frustrated many methods of file transfer.) Each log file had to be opened in MSAccess and saved to a text file through a sequence of interactions within the MSAccess application (Figure 6-2, Step 1). The MSAccess facility that exports a text file was faulty in that records would often be out of sequence in the text file, which required us to write a simple program that reordered the records according to their record ID (Step 2).

It was then necessary to write a parser to extract the high-level interactions from the copious amounts of log data and to filter out unwanted interactions (Step 3). This is where the bulk of our processing efforts were focused. We began by documenting the grammar of the log files and then constructing a parser to extract the user's command usage. Initially this was relatively straightforward – the grammar was based on the log output generated while testing the logger with very regular sequences of interactions. Once we were working with real data, however, the grammar and the parser were updated iteratively as new sequences of more complex interactions were observed.

An example will help to clarify some of these issues. An efficient series of interactions to select the **Symbol** command from the **Insert** menu consists of clicking on the **Insert** menu with the left mouse button, moving the mouse down through the menu until **Symbol** is highlighted, and then releasing the left mouse button. A more complex interaction to effect the same result could be clicking on the **File** menu with the left mouse button, moving the mouse up and down through this menu and even its one submenu, then moving into the **Edit** menu and moving up and down through it as well, moving through the **View** menu in a similar fashion and lastly into the **Insert** menu where **Symbol** is finally found and the left mouse button is then released. Note that the left mouse button was depressed the whole time from the start of the sequence until the **Symbol** command was finally reached. This lengthy interaction is consistent with a user searching multiple menus before the desired menu item is found.

As mentioned previously, interactions with other active MSWindows applications could be included in the log file and therefore the parser had to detect which log items were interactions within MSWord and filter out the others. Regrettably, switching between applications does not always produce a clean delimiter in the log file and therefore our processing sometimes required human intervention to detect where switching had taken place.

As previously mentioned, keyboard entry is captured and therefore content can be reconstructed from the log output, albeit very painstakingly because modifier and other control keys are embedded throughout the log file. Our parser filtered out keyboard input, allowing us to offer some guarantee of privacy to participants.

In addition to a parser, an analysis tool was created to incorporate status information about the user's personal interface (Figure 6-2, Part II). This tool took as input all of the user's log files as well as the text file that defined the user's personal interface (this text file is described in Section 4.1.2). For each log file a corresponding output file was generated in which each record was augmented with information about which of the three interfaces was active and which of the interfaces (minimal, personal, or full) included the particular function being invoked. In addition, the tool produced a single file containing summary statistics of all the log files produced from a given participant.

Appendix P contains excerpts of the grammar used by our parser and some sample output files.

Robustness

We encountered a substantial technical peculiarity with MSTracker in that it did not capture interactions robustly. The problem was not that particular interactions were missed and others were not, but rather MSTracker simply appeared to miss interactions randomly. When testing MSTracker slowly and methodically it seemed to capture all interactions, however, when using it in the context of regular word processing work it clearly missed some interactions. We have no explanation for this irregularity, other than speculation that it is perhaps tied to some system timing parameter.

Example: Comparing Three Ways to Print

The output is shown in tabular format as it is directly copied from an MSAccess database table.

Icon on Standard Toolbar:

Event	Name	Role	Value
MOUSE IDLE	Print	push button	
MOUSE	Print	push button	WM_LBUTTONDOWN CLIENT
MENUSTART	Menu Bar	menu bar	<unknown>
MOUSE	Print	push button	WM_LBUTTONUP CLIENT
MENUEND	Menu Bar	menu bar	<unknown>

Menu item in File Menu:

Event	Name	Role	Value
MENUSTART	Menu Bar	menu bar	<unknown>
MENUPOPUPSTART	File	Popup menu	<unknown>
MOUSE IDLE	File	menu item	
FOCUS	Close	menu item	<unknown>
MOUSE IDLE	Close	menu item	
FOCUS	Print...	menu item	<unknown>
MOUSE	Print...	menu item	WM_LBUTTONDOWN CLIENT
MOUSE IDLE	Print...	menu item	
MOUSE	Print...	menu item	WM_LBUTTONUP CLIENT
FOCUS	Print...	menu item	<unknown>
MENUPOPUPEND			
MENUEND	Menu Bar	menu bar	<unknown>

Hotkey Ctrl-P:

For some reason this particular hotkey doesn't get captured in MSTracker although other hotkeys such as Ctrl-O (Open) are recorded as follows:

Event	Name	Role	Value
KEYBOARD			<CTRL>O

The "<CTRL>O" value above appears to be a very straightforward instantiation of the Open hotkey and thus should be easy to parse. Regrettably, however, we found that even an 'o' typed normally, without any modifier key, is given a value of <CTRL>O in MSTracker's log data. So, for example, typing "hello" results in the value "hell<CTRL>O". For this reason it was impossible to parse hotkeys from MSTracker logs with any degree of accuracy.

Note that the above logs were generated while testing the logger. The application was being traversed methodically, selecting one function after another. Thus these event captures represent idealized output. In actual usage, there would often be many more events included in the logs that would have to be filtered out prior to analysis.

Goodness of Fit to Our Research Objectives

MSTracker was not well suited to our needs – the level of interaction detail recorded was excessive and requiring participants to manage the logger was problematic. OWL was our only other viable alternative at the time of the Pilot Study, however, and it did not record a substantial number of commands, leaving us with no choice other than MSTracker. It was only during the Pilot Study that we learned that MSTracker, like OWL, also suffers from incomplete data capture.

Our desire to use MSTracker forced us to switch to MSWord 2000 from MSWord 97 after we completed Study One. MSWord uses non-standard graphical elements and therefore must explicitly implement the MSAA API in order for events to be generated. We discovered that MSWord 97 does not fully implement the MSAA API, resulting in particular interface elements that were never being captured. MSWord 2000 fully implements the API and thus we thought that MSTracker would satisfy our needs if we used MSWord 2000 instead of MSWord 97.

Although privacy is always an issue, in the Pilot Study the privacy inadequacies of MSTracker were not a barrier. The participants in the Pilot Study were each well known to at least one person in our research team, which meant there was a high degree of trust that the log files would only be parsed for their command content.

Despite all the inadequacies mentioned, MSTracker was used for the Pilot Study.

6.2.3 Ergolight GUI-Tester

GUI-Tester by Ergolight is a third logger that we considered for the Pilot Study. It was not investigated nearly as thoroughly as either OWL or MSTracker, but we include it here in order to provide a complete account of our investigation of loggers.

GUI-Tester is the only commercially available software logger known to us. We learned of Ergolight through an exhibitor booth at the 2000 CHI conference. We were able to download a trial version of GUI-Tester from the World Wide Web. GUI-Tester was in its first release at that time.

Technical Details

The technical details behind GUI-Tester are almost completely unknown to us. No one at Ergolight responded to our email inquiries and so the only information we could glean is what we learned from using the application on our own.

Similar to MSTracker, GUI-Tester is a general logger for MSWindows applications. It appears to capture events sent to MSWindows – when a graphical element is selected (e.g., a frame, panel, button), the operating system is notified of the selection through an event. It may be the case that some/many graphical elements in an application do not send events. There is some notion of standard graphical elements in Windows and all such elements in an application will automatically generate operating system events. If the application includes custom graphical elements, they must be explicitly instrumented to call GUI-Tester routines to output a log item. Doing this instrumentation requires source code access to the application being logged.

General Criteria

How interactions are captured

GUI-Tester logs interactions at the level of interface interactions such as menu and toolbar button selection rather than at the function call level.

Log file format

The log file is stored in a proprietary format that can be exported to a text file.

Degree of user involvement required for use

It is unclear how much involvement, if any, is required from the user. The trial version of GUI-Tester has a setting for local users (on the current machine) and remote users (any other machine). For local users it appears that the GUI-Tester application must be launched and then the application being logged must be launched from within GUI Tester. For remote subjects it indicates that the logger can optionally be stored with the product being tested, which implies that they may be activated together. The option of working with remote users is only available in the commercial version of GUI-Tester so we were unable to evaluate how this aspect of it works.

Privacy

GUI-Tester does have privacy settings that: (1) permit data capture to be isolated to specified applications only, (2) allow keyboard input to be "encrypted", and (3) require the user to confirm that it is okay to send log files through a network connection for analysis. Their use of encryption appears peculiar in that it is described as "Record 'password style' ****** instead of real data." A true encryption would secure keyboard input such that it could not be accessed by unauthorized persons but only be decrypted and accessed by authorized persons. GUI-Tester appears to do a simple recoding of any character input into the asterisk character.

Processing log files

The GUI-Tester application has a number of analysis tools including charts that show where in the interface the user spent the majority of time. Given that we were not able to collect any substantial real usage data with GUI-Tester, it is not possible for us to properly evaluate these tools.

The actual log files appear as though they would be easy to parse because they are text files with clear delimiters between fields.

Robustness

Our experience provides some indication that GUI-Tester is likely to miss some or even most interactions unless the application being logged specifically instruments its custom widgets to generate log items. We tested GUI-Tester with MSWord 2000 and found that although the graphical elements in the default interface appear to be standard, most of them do not in fact generate events. Using the scrollbars does generate operating system level events but using the menus and toolbars does not. For this reason, GUI-Tester is probably only an appropriate logger for applications that one has implemented oneself or applications that provide source code access.

In evaluating GUI-Tester, albeit for a very limited period of time, we found it to be very difficult to use and navigate. The application suffers from some serious usability issues which is rather ironic, to say the least, given that its primary use is presumably to facilitate improved user interface design.

Example: Comparing Three Ways to Print

As mentioned above, Print invocations do not get recorded as events in MSWord. However, to provide a sense of the output generated by GUI-Tester, the respective invocations in MSWordPad[49] are provided below:

Icon on Standard Toolbar:

```
Standard; Class: ToolbarWindow32; In form: WordPadClass
```

(Note that *all* buttons on the **Standard Toolbar** generate the above record so it is not possible to distinguish the activation of the **Print** button from any of the other buttons.)

Menu item in File Menu:

```
&File; Class: Menu Bar; In form: WordPad Class
&Print… Ctrl+P; Class: Menu Bar; In form: WordPad Class
```

Hotkey Ctrl-P:

not captured

Note that the actual class names of GUI objects within MSWordPad are being generated.

Goodness of Fit to Our Research Objectives

Clearly this software did not meet our needs. Most of the commands were not being logged and we could not get source code access to instrument MSWord so it would generate loggable events.

[49] MSWordPad is a word processor that is bundled for free with the MSWindows operating system.

6.2.4 IV

The Instrumented Version of MSOffice, IV, was used in Study Two. We learned of IV during discussions that followed an invited talk we gave at Microsoft Research shortly after completing the Pilot Study. IV is logging technology that the Product Planning Team for MSOffice developed to capture users command usage in the context of their real work. Their goal was to determine command coverage in MSWord – which commands were being used and how frequently. Microsoft used IV to log a substantial number of users (500+) in the field during the 2000 calendar year and had the log files sent directly to a server. The designers had several goals for the implementation of IV: it had to be easy to install and uninstall, it could not modify users' customized files/settings/templates, and it could have only a minimal impact on users' system performance. Although not explicitly mentioned in the informal technical documentation, one would assume as well that it had to consider privacy concerns very seriously.

Technical Details

This technology works by activating internal hooks within MSOffice. These hooks are embedded more deeply in the software than what is provided through the VBA APIs. Because of this, creating a logger such as IV requires source code access. The logging hooks exist in every version of MSOffice 2000 – all copies of MSOffice are instrumented and have the capacity to produce output logs. However, to activate the logger some additional .dll files must be placed in the MSOffice directory in a user's file system and a key must be set in the Windows registry. When an MSOffice application is started it queries the registry to see if recording should occur.

General Criteria

How interactions are captured

Interactions are captured at the level of interface interactions. In addition to capturing what we defined in Study One as first-level functions such as menu and toolbar items, IV also captures second-level functions such as items selected in dialog boxes. (See Section 3.1.1 for our description of first and second-level functions.)

Log file format

IV outputs a small data file in a proprietary compressed format. One month's regular use of MSWord results in a file size that is on the order of 100KB. One way that the file size is kept so small is by encoding logging events such that each type of event is coded with a number and when the data is unpacked an English descriptor is associated with a coded log item. There is a tool that unpacks the log by outputting a text file.

Degree of user involvement required for use

There is no user involvement. Without being explicitly told, a user would not be aware of IV and the logging taking place.

Privacy

With the exception of hotkeys, IV does not record any keyboard input. So privacy is maintained very strictly.

Processing log files

Processing the IV log files is relatively straightforward. The processing steps are described below and summarized in Figure 6-3.

As mentioned above there is a tool for uncompressing the IV files into text format. This tool is launched on the command line with the log file passed as an argument which facilitates the unpacking of many log files all within a single batch script (Figure 6-3, Step 1). We converted the format of the record timestamps[50] and then merged the toggling and personalizing records that were generated separately by our modifications to the interface (see below) to form one comprehensive log file for each participant (Step 2). Finally the second-level functions that were not analyzed in Study Two were filtered out (Step 3). Because interface interactions are logged at a high level (menu, toolbar, and dialog box

[50] IV timestamps are the number of milliseconds since the client machine was booted. Luckily IV also provides a record on MSWord startup that includes the absolute time in seconds between January 1, 1970 and the last machine boot. Using all of the timing information allowed us to convert the timestamps for each data record into the absolute timestamp of calendar date and time in hours, minutes, and seconds. Using milliseconds since the last machine boot is an odd choice for a timestamp and there was never any explanation for why this format was chosen. The IV developers have indicated that absolute timestamps are now being used in the successor version of IV.

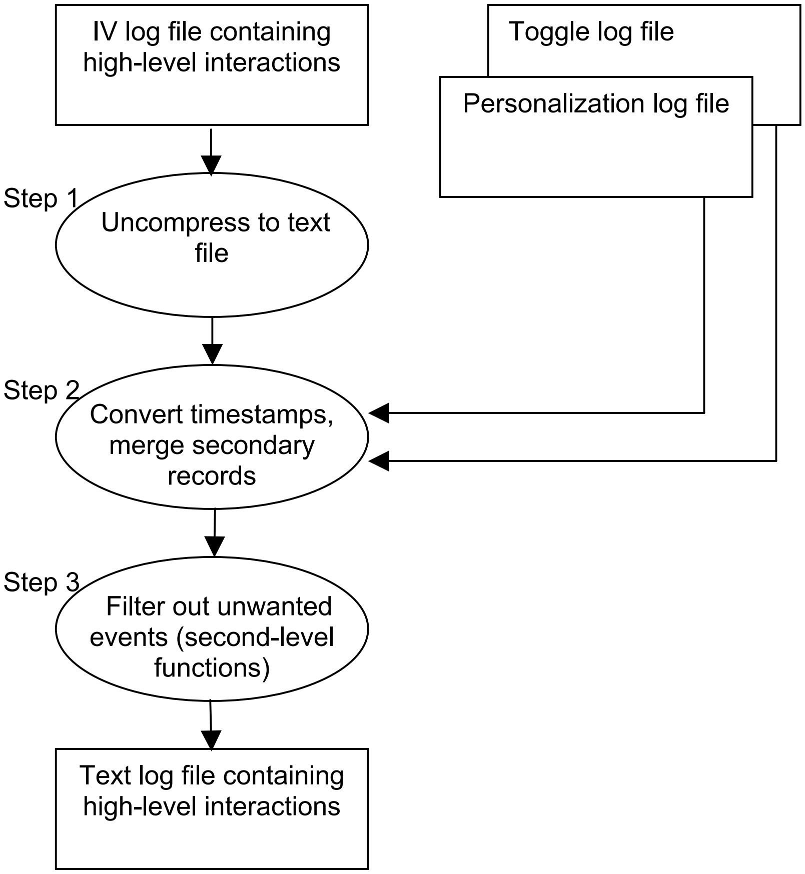

IV log file containing high-level interactions
Toggle log file
Personalization log file
Step 1
Uncompress to text file
Step 2
Convert timestamps, merge secondary records
Step 3
Filter out unwanted events (second-level functions)
Text log file containing high-level interactions

Figure 6-3: Processing IV log files.

selections), there is no grammar required. For this reason, processing the IV log files is substantially easier than processing MSTracker log files.

The analysis tool developed for the Pilot Study (Figure 6-2, Part II) was reused in Study Two with minor modifications (e.g., it only needed to accept one log file as input rather than many because with IV there was only one log file generated per participant instead of the one log file per day generated with MSTracker).

Robustness

One deficiency with IV is that it does not differentiate among custom GUI elements. In the case of our prototype, the personalizing mechanism for adding or deleting functions and the toggling mechanism for switching between interfaces are all custom GUI elements and therefore they all generate a single generic log item in IV. We were able to circumvent this deficiency by instrumenting our prototype to output timestamped records of the toggling and personalizing activity in separate log files. These were then merged with the IV log files, as previously noted.

Example: Comparing Three Ways to Print

Icon on Standard Toolbar:

```
9     2521  Standard    Print
```

Menu item File Menu:

```
40    4     File        Print
```

Hotkey Ctrl-P:

```
9     2521  Standard    Print Ctrl+P
```

Goodness of Fit to Our Research Objectives

IV provided an almost perfect fit for our needs in Study Two. The file size is very small, which enabled log files to be easily uploaded to the University of Toronto server using the basic File Transfer Protocol. Working with a text file format allowed for easy batch processing. No document content is recorded, which enabled us to assure our participants of full privacy. As is obvious from the printing example above, the logger output is clean and requires only straightforward processing.

The only missing component which would have made for more robust analysis is the notion of an active document – with the multiple interface prototype the user can have their personal interface set as the current interface in one document and the full interface set in a different document. It is not possible to determine this from the log files because IV does not capture Windows focus events.

6.2.5 Summary of Four Loggers

Table 6-1 summarizes the four loggers with respect to the general criteria.

Criteria	OWL	MSTracker	ErgoLight	IV
Captured Interactions	function call	low-level detailed interactions related to display update	menu items, toolbar buttons, scrollbars, keyboard input	menu items and toolbar buttons, hotkeys, dialog box parameters
Log File Format	text	MSAccess database	proprietary exportable to text	proprietary exportable to text
Degree of User Involvement Required	low – can easily be turned on/off, default is on	high – user required to start/stop logger and supply log filename	unknown	none
Privacy	good – only logs MSWord and no keyboard input recorded except filenames	poor – records some keyboard entry and cannot isolate logging to a single application	great – can control keyboard capture and isolate logging to a single application	good – only logs MSWord and keyboard entry not logged
Processing log files	easy	challenging – grammar required to derive semantic events from lexical events	appears easy and has some built-in tools	easy
Robustness	incomplete – missing 25% of the toolbar buttons	incomplete – randomly misses some interactions	incomplete for applications that use custom widgets unless explicitly instrumented	robust
Used in our Research		Pilot Study		Study Two

Table 6-1: Summary of the four loggers investigated with respect to the general criteria.

6.3 The Ideal Logger

From our experiences with logging technology we can now propose the characteristics of an ideal logger. The goal should be to make a general logger that will meet a wide range of research objectives. The characteristics are described below with respect to the general criteria.

How interactions are captured

The ideal logger would be a generic logger that is capable of capturing low-level and high-level interface interactions as well as underlying function calls, providing maximal information from which the desired information could be selected. Of course, if not all of this information is desired, the logger should be configurable to only capture partial information thereby resulting in smaller log files. If different event types could be tagged as such in the log file this would facilitate parsing/filtering.

In order for underlying function calls to be recorded with a generic logger, an application would need to be instrumented, and thus some form of instrumentation API would be necessary. Application developers would need to implement the API if they wanted their application to be loggable at the function call level.

Log File Format

The output file format should be text or some form of compressed text because this provides maximum post processing flexibility. If a proprietary format is used, it should be easily convertible to plain text. It should be possible to do this conversion through a batch process so that many logs can be processed at the same time from a single script.

Degree of User Involvement Required

For maximum flexibility it should be possible for the user to manage the logger manually and it should also be possible to set the logger to activate without any user involvement. If keyboard input is captured, it is necessary that there be an easy-to-use mechanism to start and stop data capture. It might also be desirable to have a facility where the user can inspect the log contents displayed in a human-readable form so that if sensitive data were captured by mistake the user could inspect and delete as required. Another reason for enabling the user to control the logger is that it might be used selectively when the user (or a researcher) is trying to record only a subset of his/her interactions, perhaps for capturing the exact sequences of interactions required to effect a certain outcome.

Privacy

Every logger should provide an option to capture keyboard input, as there are obviously situations where this data is required. When privacy is not an issue, such as in most experimental studies, then capturing keyboard input could be enabled. This of course comes with the cost of log file size and processing difficulty. When privacy is an issue, then keyboard input can be suppressed from the log files.

Providing an option for encrypting log files might be a good idea in the case of field studies where sensitive data is captured. This would make the files relatively safe during transit and only the researcher would have the decryption key necessary to access the data.

Processing log files

Any "full service" logger should be accompanied by a suite of tools that facilitate processing and analyzing the log file content. Hilbert and Redmiles [2000] categorize eight approaches and associated techniques for processing events. Each of these could be encapsulated in a tool. We summarize the proposed tools below. Hilbert and Redmiles list existing software (commercial and research prototype) that incorporates the given approaches, so readers may wish to refer to their survey for additional information.

1) **Synchronization and Searching** – a tool that allows logs to be searched interactively and that synchronizes and cross-indexes the data with other related data sources such as video or observational data, or even secondary sources of logged data.

2) **Transformation** – essentially a filter that selects events of a given type and outputs a new log file with only the selected events. This allows views to be extracted from the data. For example, one could create a view that gave only menu selection interactions. The newly generated log file could be used as input to other tools.

3) **Counts and Summary Statistics** – an aggregation tool to summarize user behaviour with respect to the frequency of command usage, the number of errors encountered, or total time spent in an application.

4) **Sequence Detection** – essentially a parser to extract sequences of interface interactions and generate the associated semantic event. This could be used for abstracting low-level interactions into interactions such as menu selection. It could also be used to abstract interactions such as menu selections into higher-level tasks such as formatting a table. The challenge here lies primarily in generating the grammar used by the parser. Determining, for example, the composition of interactions that form a task is likely a research question itself.

5) **Sequence Comparison** – related to sequence detection, this tool would detect sequences as specified in the grammar and then compare the detected sequence to an ideal sequence that results in the same semantic event. For example, one might want to compare the various actual ways in which the **Symbol** menu item is selected in practice to the more efficient way of selecting it as described in Section 6.2.2.

6) **Sequence Characterization** – this tool would take sequences as input and attempt to construct an abstract model to summarize or characterize interesting features of those sequences. If, for example, this tool was given all sequences in the data of the user selecting the **Symbol** menu item, it might detect that most sequences include a delayed hovering above the menu item before it is selected. This could signify a hesitation on the user's part, which might signal that **Symbol** is a poor label.

7) **Visualization** – a tool that would present the output from previously mentioned tools in a format that can easily be comprehended by humans.

8) **Integrated Support** – an environment that allows the composition of the various tools.

Robustness

Clearly an accurate, complete, and stable logger is desirable. If a logger is expected to be used with a wide variety of applications, and those applications are constantly undergoing changes, it may be quite difficult to achieve this goal. A trade-off between robustness and generality is likely to be necessary in this case.

6.4 Summary

After investigating four loggers we did in the end find one that met our needs almost perfectly, namely IV. It is telling that this logger is specific to MSOffice and, although Microsoft develops it, it is not available commercially. We speculate that the availability of an easily accessible (commercial or research) generic logger that will meet an array of needs is unlikely anytime in the near future. The Ergolight GUI-Tester appears to be vying for this niche, but based on our preliminary investigation it still requires substantial refinement.

The automatic log file processing that we did for our research falls mostly within what Hilbert and Redmiles [2000] categorize as transformation (selection and recoding) and summarizing (counts and frequencies). Synchronization was also carried out post logging with IV – the log data generated by the personalizing and toggling mechanism in the prototype was merged with the IV log files. In the case of MSTracker, sequence detection was required in our processing, however, the grammar was hard coded into the parser and so the tool was not at all generic. Much of the higher level processing of the IV log data, which involved mostly searching, was all done manually. For example, finding when a function was used (if at all) relative to when it was added was one manual analysis that was performed. It was expected that this would be faster than writing an additional tool to process the data because our study had a limited duration and number of participants. Had we expected a larger volume of data, we probably would have taken the time to develop a special purpose tool instead.

Through our exploration and evaluation of loggers we have learned a lot about the practical side of logging. Hilbert and Redmiles focus primarily on techniques for analyzing logged data. Having a stream of robust data is assumed as a point of departure. Our focus in this chapter has been more on how to capture robust log data in the context of a field study so it can be processed in a meaningful way. Logging issues that are especially relevant to field studies include privacy, log file size, and user management of the logger.

We argue that how a logger operates technically has a direct impact on the stream of data that is captured. For example, MSTracker captures display update events and therefore generates voluminous data. IV, on the other hand, operates through internal hooks in MSWord and so it is able to output menu and toolbar selections directly as single events. The two output streams are related in that IV data can be derived from MSTracker data by first applying a selection operator to filter out data from applications other than MSWord and then applying a sequence detection operator to abstract low-level interactions into semantic events such as menu selection. Had all other things been equal (such as robustness and degree of user involvement), our choosing IV over MSTracker could be seen in part as selecting a tool that provided partial “up front” log processing for free.

CHAPTER 7 Conclusions, Contributions, and Future Work

The main objectives of this dissertation research have been to gain a systematic understanding of users' experiences with complex software and to develop and evaluate a new interface that is derived from this understanding. These objectives have been fulfilled – two formal user studies were conducted (Study One and Study Two) as well as a pilot study that preceded Study Two, the multiple-interfaces design was conceptualized, a prototype was implemented, and it was evaluated.

We conclude this dissertation by summarizing the main contributions made in this research and by indicating where further research is still required.

7.1 Research Contributions

7.1.1 Bloat: the Objective and Subjective Dimensions

In Study One we not only determined the use and familiarity of functions, but we also heard what the users had to say about their experience of having so many functions to choose from. This initial research enabled us to further specify the notion of software bloat by defining it to include both an objective and subjective dimension. Objective bloat is the subset of features that are not used or wanted by any user. Subjective bloat is defined as the subset of functions that are not used *and* not wanted by an individual user. Subjective bloat varies from user to user, so creating a simple or basic version by eliminating functions will inconvenience most users, albeit in different ways. By extension, these findings strongly suggest that a level-structured design [Shneiderman, 1997] that provides only static predefined levels would likely frustrate most users.

7.1.2 Multiple-Interfaces Design

A new interface design has emerged from our research, namely, that of multiple interfaces. The design has been conceptualized and a proof-of-concept prototype has been implemented for the commercial word processor MSWord 2000. The novelty of this design is the *combination* of the following three design elements:

1) Multiple interfaces, one that is personalized (the personal interface), one that is the full set of functions (the full interface), and a switching mechanism between interfaces that requires only a single button click.
2) The personal interface is adaptable by the user with an easy-to-understand adaptation mechanism.
3) The personal interface begins small, and so, unless the user adds many functions, the personal interface is a minimal interface relative to the full interface.

The Pilot Study and Study Two that followed it found the multiple-interfaces design was well-liked by a significant number of users; these users took advantage of their personal interfaces by adding all frequently used functions. When compared with the adaptive interface of MSWord 2000, users reported significantly higher ratings for ease of navigating through the menus and toolbars and for their ability to learn the available features with the multiple-interfaces design. Feature-shy users also had significantly greater satisfaction and a better overall sense of control with the multiple-interfaces design.

The multiple-interfaces design is in its infancy and although promising, requires further refinement and evaluation. This future work is described in Section 7.2 below.

7.1.3 Individual differences

Our research has substantially furthered the understanding of individual differences with respect to the perception of heavily featured software. The existence of such individual differences was first proposed during a CHI 98 workshop by Kaufman and Weed [1998a] and was based on an informal focus group that included 12 users. Their work was never published and, to our knowledge, it was never extended. In our Study One we found that their concept, with some significant adjustment, did have merit. We identified three groups of users: the feature-keen, the feature-neutral, and the feature-shy, and we found that a user's ranking within these groups was independent of expertise and functions used. Of the two unbiased users who participated in the Pilot Study, the user who strongly preferred the multiple interfaces was found to be a feature-shy user and the user who was indifferent to the multiple interfaces was found to be feature-keen. This motivated us to include individual differences as an independent variable in Study Two. In the latter we found that feature-shy users did have significantly increased satisfaction while using multiple interfaces, while satisfaction reported by the feature-keen did not change significantly. A similar finding for users' reported sense of control over their software was also found. One way to interpret this finding is that through appropriate design we can positively affect one group of users without negatively affecting another group. This is a strong finding.

Based on our research, individual differences with respect to the perception of heavily featured software does appear to have construct validity. Further research is required to validate the instrument used to assess these differences and to understand how this aspect of personality relates to other well-documented personality differences[51].

7.1.4 Personalization: Adaptable versus Adaptive Interaction

Our research has broadened the understanding of personalization as a design alternative to all-in-one interfaces. Only 4 participants (20%) from Study Two ranked an all-in-one interface as their first choice, which strongly indicates that the majority of users prefer a personalized interface solution.

The comparison between adaptive and adaptable personal interfaces has been mostly theoretical to date. Our Study Two allowed us to compare one instance of each of these design alternatives in the context of a real software application with users carrying out real tasks in their own environments. Results favoured the adaptable design but the adaptive

[51] Dr. Ian Spence (Psychology) and Dr. Mark Chignell (Mechanical Industrial Engineering) have recently begun work at the University of Toronto on a Technographic Profiling Inventory. One of the goals of their research is to identify individual differences with respect to technology in general. Whereas our work has been focused on the heavily featured nature of some technology, they will be looking at many aspects of technology.

interface did have some support: 13 participants (65%) ranked adaptable first and 3 participants (15%) ranked adaptive first. Users were capable of personalizing according to their function usage and those who favoured a simplified interface were willing to take the time to personalize. The finding of a 4-to-1 preference for an adaptable design that gave users full control, over an adaptive design that gave users minimal control, provides strong evidence that if design is going to err in one direction, it should be that users are given too much control rather than too little.

The adaptive menus in MSWord 2000 are to our knowledge the first commercial implementation of an adaptive design. There have been no evaluations of this design reported in the literature and even within Microsoft itself there has been minimal evaluation carried out[52]. Our Study Two is the most extensive evaluation of this design reported in the literature [McGrenere, Baecker, & Booth, 2002].

7.1.5 Methodology

The methodology used in our research is novel in several ways.

Throughout this research the needs of the user have been paramount – we have deliberately avoided a technology-centered approach to design. We began with a broad, comprehensive approach to understanding the problem of feature bloat from the users' perspective and arriving at an interface design. The inclusion of Study One, which was a thorough and systematic evaluation of a relatively large number of users, makes our work unique. In our Pilot Study we then evaluated the merits of a personalized interface independent of a personalizing mechanism using Wizard of Oz methodology. At each stage of the research we listened to our users and this factored heavily in the resulting multiple-interfaces design.

The quasi-experimental design developed and used in Study Two was novel. Evaluating and comparing personalization technologies is fundamentally challenging in that the inherent personal nature of the technologies means that there are dependencies on the context of use, the tasks undertaken, and on individual user characteristics. Traditional laboratory

[52] Personal conversation with David Caulton, MSOffice Usability Team, May 12, 2000.

experimental design and field evaluation techniques both have limitations and advantages for the unique problem of evaluating personalization. Our Study Two utilized an innovative field study design that also included an experimental component. The electronic interconnectedness between the researcher and the participants (email correspondence, online questionnaires and personal schedules, and the transfer of logging data) allowed us to conduct the study in the field and collect a significant amount of data without being overly intrusive. Recruiting participants who were already MSWord 2000 users made it possible for us to find a sufficient number of participants and enabled some comparison between the two interface designs[53].

7.1.6 Cross-disciplinary Research Approach

Related to methodology, our research approach has been cross-disciplinary; one of the innovative aspects of this work has been the bringing together of Sociology and Computer Science in truly collaborative research. Computer scientists have an understanding of how things work technically, and are capable of theorizing and thinking about how things could work differently. Sociological imagination, on the other hand, is grounded in the social setting, the relationships, and the context. In our research the two disciplines came together to discover how people experience complex software and to use the findings to inform future interface design, that of the complexity of an everyday software application. The disciplines animate each other and the result is what Moore has called the "cross-disciplinary imagination"[54]. Both disciplines are essential; both are hard. But then so are real world problems.

Study One was strongly grounded in sociological methods, but was deeply informed by Computer Science expertise. The detailed design and implementation of the prototype for the Pilot Study and Study Two was Computer Science research, but remained grounded in the earlier work. For the evaluation of our multiple-interfaces design in Study Two sociological

[53] We are quite certain that had we tried to recruit users who had not yet upgraded to MSWord 2000 but who were willing to have us do the upgrade, we would likely not have found sufficient participants.

[54] Personal communication with Dr. Gale Moore, November 19, 2001.

methods were used once again. Sociology can play an important role in setting the problem and in helping to understand and reveal the experiences and practices of those using technology in a variety of social contexts. Computer Science provides an understanding of the technical possibilities, the scope for possible solutions, and the realization of those solutions. Working with other "non-technical" disciplines is an emerging way of doing research. Computer scientists now regularly work with psychologists; their relationship with sociologists is, to date, still limited.

7.1.7 Software Logging in Practice

Our considerable experience with software logging technology has enabled us to specify issues pertaining to the practical side of logging which include privacy, user management, and log file format and file size. In addition, we noted that how a logger is implemented and where it resides with respect to the software architecture of the application being logged has significant implications for not only the practical side of logging but also the stream of logging data produced. Our own research would have been greatly facilitated by a generic software logger and a comprehensive set of data analysis tools. We are likely not alone – HCI researchers in general would benefit from such a logger. Although a generic logger may not be fully achievable, there is still much room for improvement over the current solutions available. We have provided some suggestions in this direction.

7.2 Future Work

7.2.1 Further Evaluation of the Multiple-Interfaces Design

The best way to ensure that the results of our user studies are robust is to replicate our work. As such, another study identical to that of Study Two could be conducted and the results could then be compared. In addition, conducting other related studies, perhaps variations on the Study Two design, would provide further insight into users' experience of multiple-interfaces designs. Three potential studies are described in this section:

Longitudinal Study

In order to mitigate the effect of multiple treatments a longer study could be conducted. Users could use the multiple-interfaces design for an extended period of time (perhaps 6 months) and then return to the adaptive design for an equal period of time. This would likely make the users feel as familiar with the multiple-interfaces design as with the adaptive or all-in-one designs. One challenge with such a longitudinal design, however, is that it would likely suffer a greater loss of participants and would therefore have to be more qualitative in nature rather than quantitative.[55]

Evaluation of Novice Users

Study One revealed the existence of novice users who had a negative experience of the functionality in MSWord. Yet, we did not include novice users in either the Pilot Study or Study Two. We felt that all levels of expertise could not be adequately evaluated within the same study design and we chose to focus on intermediate and expert users. Needless to say, to make our work more complete the multiple-interfaces design should be evaluated with novice users. This would also provide a valuable extension to the Training Wheels research [Carroll & Carrithers, 1984a, 1984b] that was conducted nearly two decades ago but never advanced. We suspect that novice users would benefit from a multiple-interfaces design but rather than having an initial personal interface that includes only six functions as was done in Study Two, a more substantial base set of functions would likely be beneficial, for example, the top 20% of the most frequently used functions.

In-depth Evaluation of Learning

One of our interests from the outset of this research has been to understand the impact of having many features to choose from on the user's ability to learn those features. The results from Study Two showed that users reported the multiple-interfaces design to be significantly easier for learning the available features than the adaptive version, but the interview data indicated that the majority of users feel that the all-in-one design supported learning better

[55] Our Study Two was approximately 6 to 7 weeks in length. Had it lasted longer, we would likely have lost 2 of our participants. These participants left their companies the week following their completion of the study. These job changes were not known to the participants when they began the study.

than multiple interfaces. Interestingly, 4 users thought that having access to the full interface was sufficient for learning or perhaps better because it partitions the learning space. A laboratory-style experiment would likely resolve some of these questions.

7.2.2 Extending the Multiple-Interfaces Concept

Beyond performing additional evaluation of our current two-interfaces implementation, extensions to our design need to be explored. When the multiple interfaces design was conceptualized in Study One we noted two key parameters to the design space: the number of interfaces included and the basis for differentiating the interfaces.

Our research has not clarified how many interfaces the user can manage. In the Pilot Study there were three interfaces but the minimal interface only appeared to have value for 1 of the 4 users and did cause some confusion for another user. We ask whether having more than two interfaces was the problem or was it the particular choice of interfaces? It could be that the particular set of functions in the minimal interface was the problem or that if one uses reduced functionality sets as the basis for the interfaces, then two interfaces is all that is needed.

One user in Study Two wanted to be able to construct more than one personal interface and gave the example of having different interfaces for different main tasks. So here she is identifying task-based differentiation of the interfaces. One has to ask whether or not this user would actually have been willing to spend the time to set up more than one personal interface? Would others?

One can also imagine a relatively different basis for the interfaces. A new version of an application could include the interface of its predecessor version; for example, MSWord 2000 could include the MSWord 97 interface. Some users delay upgrading their software because of the time required to learn a new version. By allowing users to continue to work in the old interface while also accessing the new interface, they would be able to transition at a self-directed pace. Similarly, multiple interfaces might be used to provide a competitor's interface in the hopes of attracting new customers. For example, MSWord could offer the interface to a word processor such as Word Perfect, in order to support users gradually transitioning to the Microsoft product. Clearly there are differences between two interfaces beyond menus and toolbars, so this would need further thought.

7.2.3 Exploring the Boundary Between Adaptable and Adaptive Interfaces

How can we assist the user to make appropriate adaptations? In terms of our prototype implementation of the multiple-interfaces design, one potential way of assisting personalization would be to add a mechanism that provides usage information and allows the user to directly add features that have been used frequently or recently. For example, one could incorporate a mechanism similar to Kaasten and Greenberg's [2001] revisitation system. This would move the design in the direction of user-assisted personalization, where the user has ultimate control but also benefits from user-modeling technology.

In general, the boundary between user control and intelligent assistance is ripe for further investigation; relatively little has been done. There has been minimal research into understanding customization in applications with graphical user interfaces. Our work makes a contribution in this area, but what about having users control more complex customizations? How should customization facilities be designed to accommodate this? Researchers investigating Intelligent User Interfaces have acknowledged the need for more user control but many questions remain open. For example: What kind of information will assist the user in adapting the interface? How should this information be presented? What kind of adaptations, if any, would the user be willing to have the system carry out on its own? How should these adaptations take place, e.g., should the user be notified? And if so, how?

7.2.4 The Broader Problems of Complex Software

Our multiple-interfaces prototype only partially met the challenge of supporting the user's creation of a personal interface. It provided a mechanism by which the user selected items from the full interface to add to the personal interface, however, the fundamental question this design did not address is: What functions *should* I add to my personal interface? This question targets the challenging issue of determining which functions should be learned and used. Previous research has shown that users learn mostly through exploration in the context of a given task [e.g., Rieman, 1996] and these results are supported by the findings of our

Study One. The learning approach of choice thus tends to be what we call “just-in-time exploration.” One challenge is that novices tend to be more cautious than experts fearing that they might “break” something.

Taking this all into consideration, we believe that some form of “exploratory mode” would significantly benefit users of all levels of expertise. When users enter the exploratory mode they are given an opportunity to explore the interface with impunity. Some possibilities of what this might include are:

- Additional information about the features in the interface, for example, when a feature is selected an information sheet pops up that explains the feature in layman’s terms and expert’s terms and an example is given, and perhaps a list of related features is given with hyperlinks provided.
- A video demonstration of the feature in use[56].
- The ability to try the feature on a “dummy” document.
- The ability to load a document and query which features were used to construct a certain aspect of the document. Perhaps this would involve a selection tool that allows for the selection of any region in the document. Once selected, a listing of all features that were applied to that particular region of the document is given.
- The ability to try the feature on the current document and when the exploratory mode is exited, the choice is given to keep the changes made or discard them altogether.

An exploratory mode could be combined effectively with our multiple-interfaces model. Within this mode, the user could be given the opportunity of adding the feature to his/her personal interface.

[56] Expresto Software Corp. has developed technology that allows for “the efficient creation and just-in-time delivery of visual FAQs, user guides, ‘how-to’ documentation, searchable references, e-learning material, and product tours.” [Expresto, 2001]

7.3 A Final Word

The concept of multiple interfaces is compelling. We have shown that a basic model with two interfaces, one of which is personalizable by the user, can improve how a significant number of users experience a complex productivity application. Some improvements of this model do remain, such as streamlining the process for constructing the personal interface. Investigation in this area will result in a major contribution to personalization research. But it should not end at that point, as there is much potential beyond this first model. To start, one can ask in what other ways this model can be applied. We have begun to address this question by identifying the use of multiple interfaces as a means to support users transitioning from one version of an application to the next. In addition, multiple interfaces could be incorporated into an exploratory mode that supports users' learning of an application through their preferred method of just-in-time exploration. Beyond these, there are undoubtedly many other multiple-interface design possibilities.

References

Baecker, R., Booth, K., Jovicic, S., McGrenere, J., & Moore, G. (2000). Reducing the gap between what users know and what they need to know. *Proceeding of ACM Conference on Universal Usability 2000*, 17 - 23.

Boehm, B.W. (1988). A spiral model of software development and enhancement. *IEEE Computer, 21(2),* 61 - 72.

Campbell, D.T., & Stanley, J.C. (1972). *Experimental and quasi-experimental designs for research*. Chicago, IL: Rand McNally & Company.

Carroll, J., & Carrithers, C. (1984a). Blocking learner error states in a training-wheels system. *Human Factors, 26(4),* 377 - 389.

Carroll, J., & Carrithers, C. (1984b). Training wheels in a user interface. *Communications of the ACM*, *27(8),* 800 - 806.

Carroll, J., & Mack, R. (1984). Learning to use a word processor: By doing, by thinking, and by knowing. In J. Thomas & M. Schneider (Eds.), *Human factors in computer systems* (pp. 278 - 297). Norwood, NJ: Ablex. Publishing.

Computer Science and Telecommunications Board, National Research Council (1997). *More than screen deep: Toward every-citizen interfaces to the nation's information infrastructure.* Washington, DC: National Academy Press.

Constantine, L.L. (1995). *Constantine on peopleware*. Englewood Cliffs, NJ: Prentice Hall.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis, IN: SAMS.

Cote-Muñoz, J.A. (1993). AIDA - An adaptive system for interactive drafting and CAD applications. In M. Schneider-Hufschmidt, T. Kuhme, & U. Malinowski (Eds.), *Adaptive user interfaces: principles and practice* (pp. 225 - 240). North Holland: Elsevier Science Publishers B.V.

Csinger, A., Booth, K.S., & Poole, D.L. (1995). AI meets authoring: User models for intelligent multimedia. *Artificial Intelligence Review Journal, Special Issue on Integration of Language and Vision Processing, 8*, 447 - 468.

Davis, J., Dye, J., Johnson, N., & Bell, S. (1999). Microsoft Usability Report, IDIS #5500.

Dieterich, H., Malinowski, U., Kühme, T., & Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. In M. Schneider-Hufschmidt, T. Kuhme & U. Malinowski (Eds.), *Adaptive user interfaces: Principles and practice* (pp. 13 - 48). North Holland: Elsevier Science Publishers B.V.

Dillon, A. (2001, March 22). Invited Speaker for the Knowledge Media Design Institute Lecture Series on Humanizing Technology [On-line]. http://www.kmdi.org/events.htm

Do computers have to be hard to use? Complex, volatile, frustrating; There must be a simpler way (1998, May 28). *New York Times*, 1 - 5.

Dryer, D.C. (1997). Wizards, guides, and beyond: Rational and empirical methods for selecting optimal intelligent user interface agents. *Proceedings of ACM IUI 97*, 265 - 268.

Expresto, Corp. (2001, October 16). [On-line]. http://www.expresto.com/explain_anything/index.html

Fischer, G. (1993). Shared knowledge in cooperative problem-solving systems – integrating adaptive and adaptable components. In M. Schneider-Hufschmidt, T. Kuhme, & U. Malinowski (Eds.), *Adaptive user interfaces: principles and practice* (pp. 49 - 68). North Holland: Elsevier Science Publishers B.V.

Fisher, B., & Dill, J. (1999). CZWeb: A web-based workspace for media-rich communication and decision evolution [On-line]. http://www.cs.ubc.ca/~fisher/czweb99.pdf

Franzke, M. (1995). Turning research into practice: Characteristics of display-based interaction. *Proceedings of ACM CHI'95*, 421 - 428.

Franzke, M., & Rieman, J. (1993). Natural training wheels: Learning and transfer between two versions of a computer application. *Proceedings of Vienna Conference VCHCI'93*, 317 - 328.

Gantt, M., & Nardi, B. (1992). Gardeners and gurus: Patterns of cooperation among CAD users. *Proceedings of ACM CHI'92*, 107 - 117.

Gaudin, S. & Nash, K.S. (1998, August 10). Retail user groups tackle 'bloatware'. *Computer World 32(32)*, 1,74.

Gibbs, W. (1997, July). Taking computers to task. *Scientific American*, 82 - 89.

Gong, G., & Salvendy, G. (1995). An approach to the design of a skill adaptive interface. *International Journal of Human-Computer Interaction, 7(4),* 365 - 383.

Greenberg, S. (1993). *The computer user as toolsmith: The use, reuse, and organization of computer-based tools*. Cambridge: Cambridge University Press.

Greenberg, S. & Witten, I.H. (1985). Adaptive personalized interfaces - A question of viability. *Behaviour and Information Technology, 4(1)*, 31 - 45.

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

Hanson, S.J., Kraut, R.E., & Farber, J.M. (1984). Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems, 2(1)*, 42 - 57.

Hilbert, D.M. & Redmiles, D. F. (2000). Extracting Usability Information from User Interface Events. *ACM Computing Surveys, 32(4)*, 384 - 421.

Hirsh, H., Basu C. & Davison, B.D. (2000). Learning to personalize. *Communications of the ACM*, *43(8),* 102 - 106.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere Project: Bayesian user modeling for inferring the goals and needs of software users. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 256 - 265.

Howes, A. & Payne, S.J. (1990). Supporting exploratory learning. *Proceedings of IFIP INTERACT '90: Human-Computer Interaction,* 881 - 885.

Hsi, I. & Potts, C. (2000). Studying the evolution and enhancement of software features. *Proceedings of the International Conference on Software Maintenance*, 143 - 151.

I see you're going to retire, Clippit (2001, April 15). *The Detroit Free Press* [On-line]. http://www.freep.com/money/tech/q1tpick15b.htm

Innocent, P.R. (1982). Towards self-adaptive interface systems. *International Journal of Man-Machine Studies, 16(3),* 287 - 299.

Johnson, J., Roberts, T., Verplank, W., Smith, D., Irby, C., Beard, M., & Mackey, K. (1989). The Xerox Star: A retrospective. *IEEE Computer 22(9)*, 11 - 29.

Kaasten, S. & Greenberg, S. (2001). Integrating back, history and bookmarks in web browsers. *Extended Abstracts of ACM CHI 2001*, 379 - 380.

Kaufman, L. & Weed, B. (1998a). User interfaces for computers – Too much of a good thing? Identifying and resolving bloat in the user interface. *Conference Summary, ACM CHI 98,* 207 - 208.

Kaufman, L. & Weed, B. (1998b). Too much of a good thing? Identifying and resolving bloat in the user interface: A CHI 98 workshop. *ACM SIGCHI Bulletin, 30(4),* 46 - 47.

Kesterton, M. (1998, May 20). Social Studies. *The Globe and Mail*, A20.

Koutsofios, E. (1996). Editing pictures with lefty [On-line]. http://www.research.att.com/sw/tools/graphviz/leftyguide.pdf

Kozierok, R. & Maes, P. (1993). A learning interface agent for scheduling meetings. *Proceedings of ACM IUI '93*, 81 - 88.

Krogsœter, M., Oppermann, R., & Thomas, C. (1994). A user interface integrating adaptability and adaptivity. In R. Oppermann (Ed.), *Adaptive user support: Ergonomic design of manually and automatically adaptable software* (pp. 97 - 124). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Kühme, T. (1993) A user-centered approach to adaptive interfaces. *Proceedings of ACM IUI '93*, 243 - 245.

Kurtenbach, G. & Buxton, W. (1994). User learning and performance with marking menus. *Proceedings of ACM CHI'94*, 258 - 264.

Kurtenbach, G. Fitzmaurice, G.W., Owen, R., & Baudel T. (1999). The hotbox: Efficient access to a large number of menu-items. *Proceedings of ACM CHI 99*, 231 - 237.

Landauer, T. (1997). Behavioral research methods in human-computer interaction. In M.G. Helander, T.K. Landauer, & P.V. Prabhu, (Eds.), *Handbook of human-computer interaction* (2nd Ed.) (pp. 203 - 227). Amsterdam: Elsevier Science B.V.

Linton, F., Joy, D. & Schaefer, P. (1999). Building user and expert models by long-term observation of application usage. *Proceedings of the Seventh International Conference on User Modeling*, 129 - 138.

Linton, F., Joy, D., Schaefer, P., & Charron, A. (2000). OWL: A recommender system for organization-wide learning. *Educational Technology & Society, 3(1)* [On-line]. http://ifets.ieee.org/periodical/issues.html

Mack, R.L., Lewis, C.H., & Carroll, J.M. (1983). Learning to use word processors: Problems and prospects. *ACM Transaction on Office Information Systems, 1(3),* 254 - 271.

Mackay, W. E. (1990). Patterns of sharing customizable software. *Proceedings of ACM CSCW'90*, 209 - 221.

Mackay, W. E. (1991). Triggers and barriers to customizing software. *Proceedings of ACM CHI'91*, 153 - 160.

MacLean, A., Carter, K., Lovstrand, L, & Moran, T. (1990). User-tailorable systems: Pressing the issues with buttons. *Proceedings of ACM CHI'90*, 175 - 182.

Malinowski, U., Kühme, T., Dieterich, H., & Schneider-Hufschmidt, M. (1993). Computer-aided adaptation of user interfaces with menus and dialog boxes. *Proceedings of the Fifth Conference on Human-Computer Interaction*, 122 - 127.

Manber, U., Patel, A., & Robison, J. (2000). Experience with personalization on Yahoo! *Communications of the ACM*, *43(8),* 35 - 39.

Maybury, M.T. & Wahlster, W. (1999). *Readings in intelligent user interfaces*. San Francisco, CA: Morgan-Kaufmann Publishers, Inc.

McGrath, J. (1995). Methodology matters: Doing research in the behavioral and social sciences. In R. Baecker, J. Grudin, W. Buxton, & S. Greenberg (Eds.), *Readings in Human-Computer Interaction: Toward the Year 2000* (pp. 151 - 169). Los Altos, CA: Morgan-Kaufmann Publishers Inc.

McGrenere, J., Baecker, R., & Booth, K.S. (1999). *Learning to use complex computer technology: The importance of user interface design.* Technical Report CSRG-403, University of Toronto, Department of Computer Science.

McGrenere, J., Baecker, R., & Booth, K.S. (2002). An evaluation of a multiple interface design solution for bloated software. Accepted for publication at *ACM CHI 2002*.

McGrenere, J. & Moore, G. (2000). Are we all in the same "bloat"? *Proceedings of Graphics Interface 2000*, 187 - 196.

Menkhaus, G. (2001). Architecture for client-independent Web-based applications. Proceedings of Technology of Object-Oreinted Languages and Systems, 2001, TOOLS 38, 32-40.

Microsoft Office 2000 (2000, February 8). *Products enhancements guide* [On-line]. http://www.microsoft.com/Office/evaluation/ofcpeg.htm

Miller, J.R., Sullivan, J.W., & Tyler, S.W. (1991). Introduction. In J.W. Sullivan & S.W. Tyler (Eds.), *Intelligent user interfaces* (pp. 1 - 10). New York, NY: ACM Press.

Munk, N. (1996, October 21). Technology for technology's sake. *Forbes*, 280 - 288.

Myers, B.A., Potosnak, K., Wolf, R., & Graham, C. (1993). Heuristics in real user interfaces. *Proceedings of ACM InterCHI'93*, 304 - 307.

Nilsen, E., Jong, H., Olson, J., Biolsi, K., Reuter, H., & Mutter, S. (1993). The growth of software skill: A longitudinal look at learning & performance. *Proceedings of ACM InterCHI'93*, 149 - 156.

Norman, Don. (1998). *The invisible computer*. Cambridge, MA: MIT Press, 80.

Norman, Kent. (1991). *The psychology of menu selection: Designing cognitive control for the human/computer interface.* Norwood, NJ: Ablex Publishing Corporation.

Online Computing Dictionary (2001, October 16). [On-line]. http://www.instantweb.com/

Page, S.R., Johnsgard, T.J., Albert, U., & Allen C.D. (1996). User customization of a word processor. *Proceedings of ACM CHI 96*, 340 - 346.

Quintana, C., Carra, A., Krajcik, J., & Soloway, E. (2001). Learner-centered design: Reflections and new directions. In J. Carroll (Ed.), *HCI in the New Millennium.* Reading, MA: Addison-Wesley Publishing.

Raskin, J. (1997). Looking for a humane interface: Will computers ever become easy to use? *Communications of the ACM, 40(2)*, 98 - 101.

References on Zipf's Law (2001, October 16). [On-line]. http://linkage.rockefeller.edu/wli/zipf/

Resnick, P. & Varian, H.R. (1997). Recommender systems. *Communications of the ACM, 40(3),* 56 - 58.

Riecken, D. (2000). Personalized views of personalization. *Communications of the ACM, 43(8),* 27 - 28.

Rieman, J. (1996). A field study of exploratory learning strategies. *ACM Transactions on Computer-Human Interaction, 3(3),* 189 - 218.

Rich, C. & Sidner, C.L. (1996). Adding a collaborative agent to graphical user interfaces. *Proceedings of ACM UIST 96*, 21 - 30.

Santos, J.R. (1999). Cronbach's alpha: A tool for assessing the reliability of scales. *Journal of Extension*, *37(2)* [On-line]. http://www.joe.org/joe/1999april/tt3.html

Sears, A., & Shneiderman, B. (1994). Split menus: Effectively using selection frequency to organize menus. *ACM Transactions on Computer Human Interaction, 1(1)*, 27-51.

Seffah, A., Radhakrishnan, T., & Canals, G. (2001, September 11). Multiple user interfaces over the Internet: Design and applications trends. Workshop 6, HCI IHM'2001 [On-line]. http://www.ihm-hci2001.org/

Shneiderman, B. (1995). Looking for the bright side of user interface agents. *ACM Interactions, 2(1)*, 13 - 15.

Shneiderman, B. (1997a). *Designing the user interface: Strategies for effective human-computer interaction* (3rd ed.). Reading, MA: Addison-Wesley Publishing.

Shneiderman, B., (1997b). Direct manipulation for comprehensible, predictable and controllable user interfaces. *Proceedings of ACM IUI 97*, 33 - 39.

Shneiderman, B., & Maes, P. (1997). Direct manipulation vs. interface agents: Excerpts from debates at IUI 97 and CHI 97. *ACM Interactions, 4(6)*, 42 - 61.

Snell, J. (1998, January). Walking the browser tightrope. *Macworld* [On-line]. http://www.macworld.com/1998/01/features/4134.html

Soloway, E., Guzdial, M. & Hay, K.E. (1994). Learner-centered design the challenge for HCI in the 21st century. *ACM Interactions*, *1(2),* 36 - 48.

SPSS Frequently Asked Questions: What does Cronbach's Alpha Mean? (2001, October 16). UCLA Academic Technology Services [On-line]. http://www.ats.ucla.edu/stat/spss/faq/alpha.html

Stephanidis, C., Karagiannidis, C., & Koumpis, A. (1997). Decision making in intelligent user interfaces. *Proceedings of ACM IUI 97*, 195 - 202.

Svendsen, G.B. (1991). The influence of interface style on problem solving. *International Journal of Man-Machine Studies, 35(3),* 379 - 397.

Tapia, M.A., & Kurtenbach, G. (1995). Some design refinements and principles on the appearance and behavior of marking menus dialogues. *Proceedings of ACM UIST'95*, 189 - 195.

Tauscher, L., & Greenberg, S. (1997). How people revisit web pages: Empirical findings and implications for the design of history systems. *International Journal of Human-Computer Studies*, *47(1),* 97 - 137.

Thomas, C.G. & Krogsœter, M. (1993). An adaptive environment for the user interface of Excel. *Proceedings of ACM IUI '93*, 123 - 130.

Trudel, C.I. & Payne, S.J. (1995). Reflection and goal management in exploratory learning. *International Journal of Human-Computer Studies, 42(3)*, 307 - 339.

Tyler, S.W., & Treu, S. (1989). An interface architecture to provide adaptive task-specific context for the user. *International Journal of Man-Machine Studies, 30(3)*, 303 - 32.

Yahoo! (2001, October 16). [On-line]. http://www.yahoo.com

Appendix A: MSWord 2000 Adaptive Menus

This appendix displays the “short” and “long” menus in MSWord 2000. Three screen captures provide an example of the **Text Box** item being selected from the “long” **Insert** menu which then causes the **Text Box** item to appear in the “short” **Insert** menu.

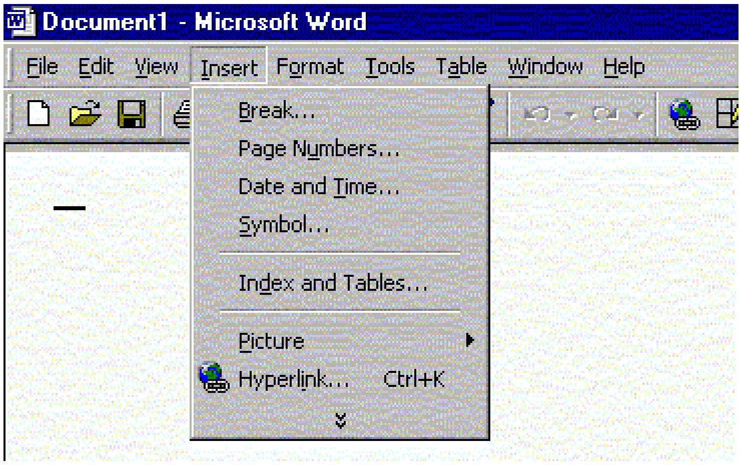


Figure A-1: “Short” ***Insert*** *menu before any menu items have been added. By clicking on the double-down-arrow icon at the bottom or by hovering in this menu for a few seconds, the “long”* ***Insert*** *menu will then be displayed.*

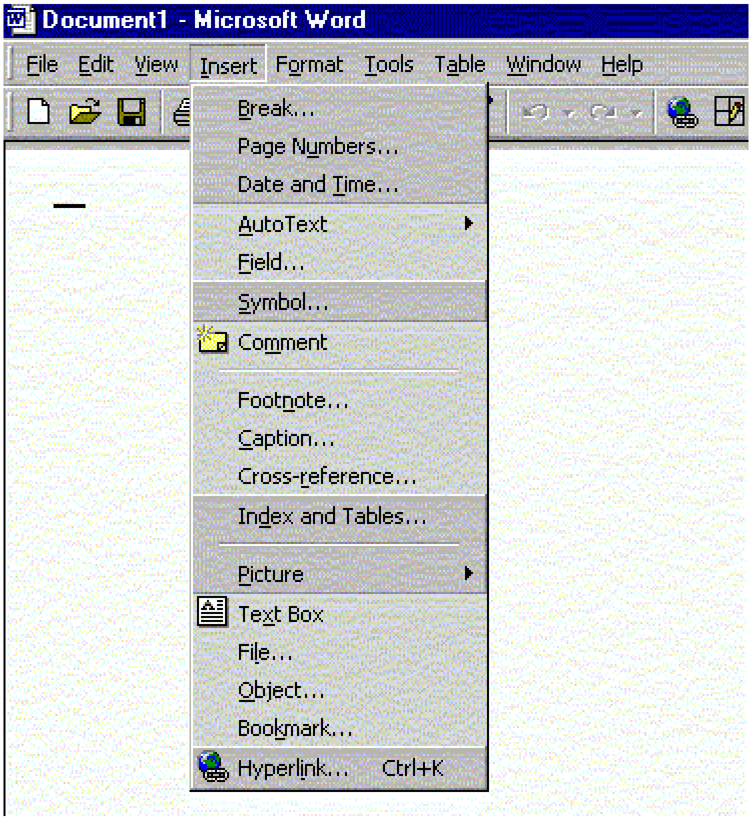


Figure A-2: "Long" **Insert** *menu showing all menu items. The items in dark gray are those that also appear in the "short" menu. Although not shown, the user selects* **Text Box** *which is the 5th item from the bottom.*

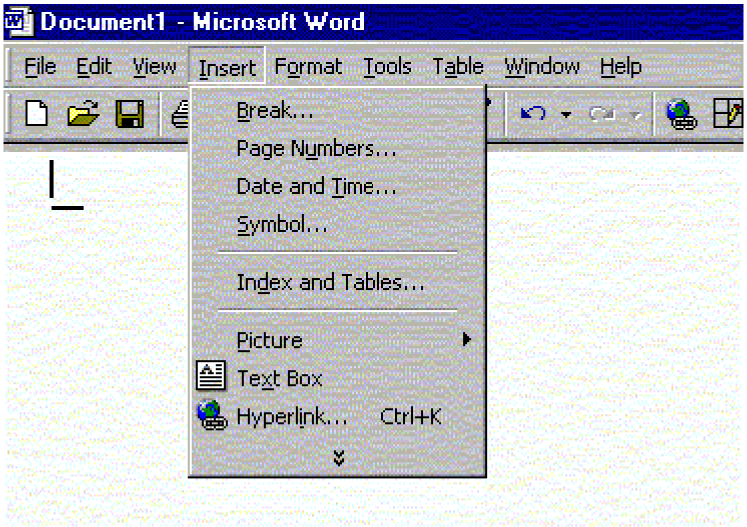


Figure A-3: **Text Box** *now appears in the"short"* **Insert** *menu. After some period of non-use* **Text Box** *will be removed from the "short" menu but will always be available in the "long" menu.*

Appendix B: Study One Counting in Dialog Boxes

The rules for counting functions in the dialog boxes are significantly more complicated than counting first-level functions. The general heuristics are given in the table below and the exceptions then follow.

Widget	Count
tabbed notebook	– each tab counts as one – for objects that appear the same across multiple tabs (within a single tabbed notebook), only count them once
text input box	– counts as one
selection widget (e.g., drop down list box, scrollable list box, list box: the number of alternatives is unbounded or very large)	– counts as one (regardless of the number of items in selection)
radio button widget	– counts as one (regardless of the number of buttons)
check box	– counts as one
button	– counts as one
Cancel/Close/Help button	– not counted
Wizard help button	– counts as one
OK button	– may or may not be counted as one – see exceptions below
reset/default button	– counts as one
button that brings up 2nd-level dialog boxes or menus	– counts as one
the same widget that appears in more than one dialog box	– counts as one each time it appears unless the dialog boxes are identical[57] in which case it is counted in full the first time and as one for all subsequent identical dialog boxes.
previews	– not counted

[57] Identical dialog boxes are ones that have exactly the same widgets (e.g. File-Save As and File-Save As HTML) or nearly the same (e.g., File-Open and File-Save As).

Exceptions:

1. **button that "executes" –** This is a button that triggers something to happen based on the status of other widgets in the dialog box. The OK button is such a button. This type of button is only counted as one if it is possible to select this button without doing anything else (making a selection, entering text into a box, etc.) and for the execution to be *meaningful*. Another way to think of this is that the default settings of the widgets in the dialog box are going to be left unchanged by the user most of the time. For example, the Open button in the Open dialog box does not get counted because it will always open the first document in the document list – the one that is highlighted by default. This may be what the user desires some of the time. However, for most users who have multiple documents in a directory, documents other than the one listed first will be desired. But the OK button on the New dialog box does count as one because the Blank Document is highlighted by default and most of the time this is what users will be creating and so they will only need to hit the OK button.

2. **Multiple widgets that "go together" or are "dependent" –** If one or more widgets are grouped together in the sense that you *must* manipulate all the widgets in the group in order for the interaction to be *meaningful* then the group of widgets is only counted as one. For example, on the Print dialog box, toggling the "pages" radio button requires the user to enter the desired page range in the text box. These two together only count as one. (If there was a meaningful default provided in this text box for the page range and the user had the *choice* of inputting a range or not, then these two widgets would *each* be counted separately.)

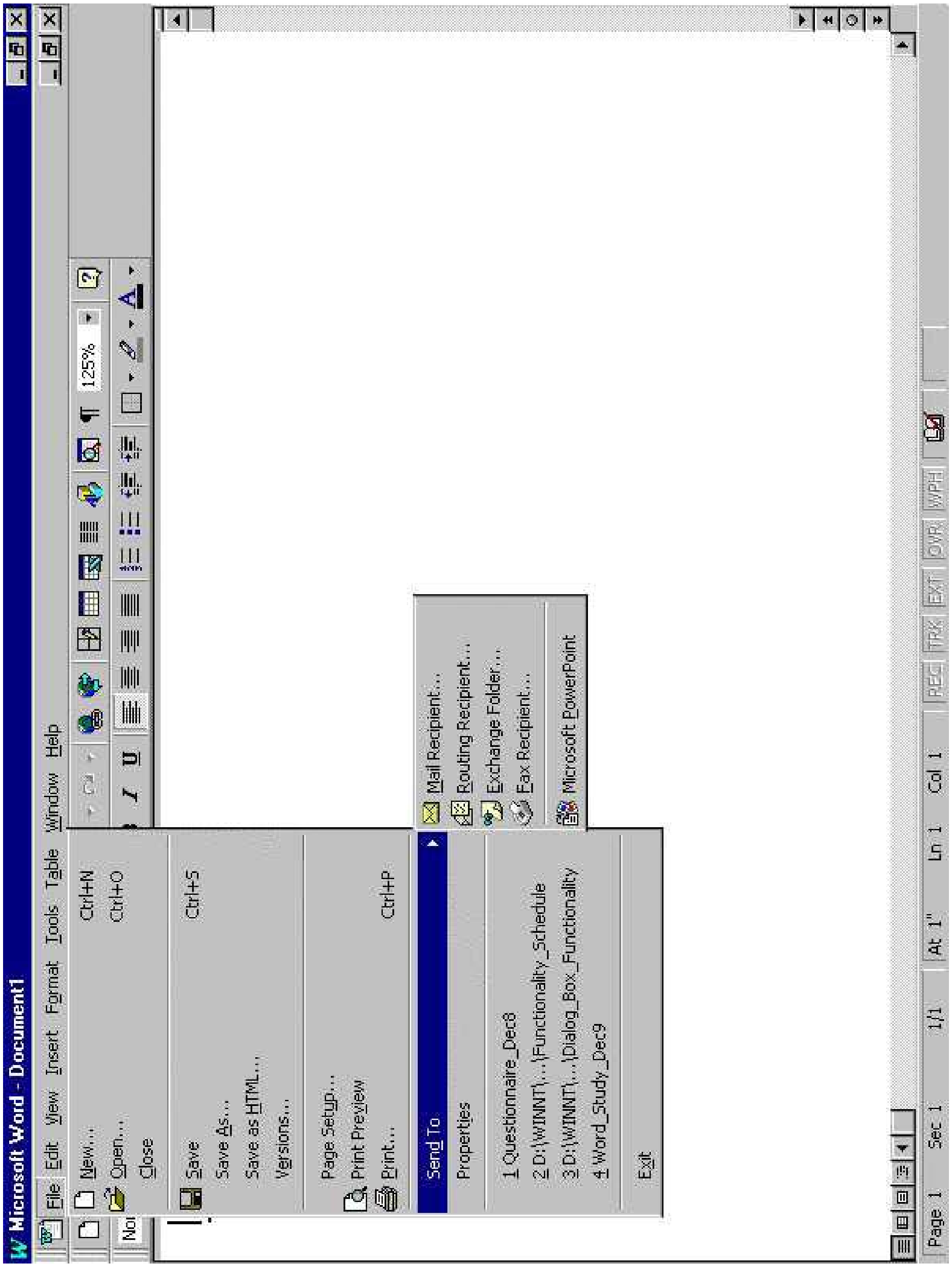
3. **"forms" –** Forms are a group of text boxes that allow the user to input data that describes their document or some aspect of their document. Example, the Summary tab in the Properties dialog box (off the File menu). All of these boxes together within a single dialog box (or tab in a tabbed notebook) count as one.

4. **displaying status information** – if the primary purpose of a dialog box is to display status information then the displaying of this information can count as one. E.g., Word Count (from the tools menu) counts the number of words, paragraphs, etc.

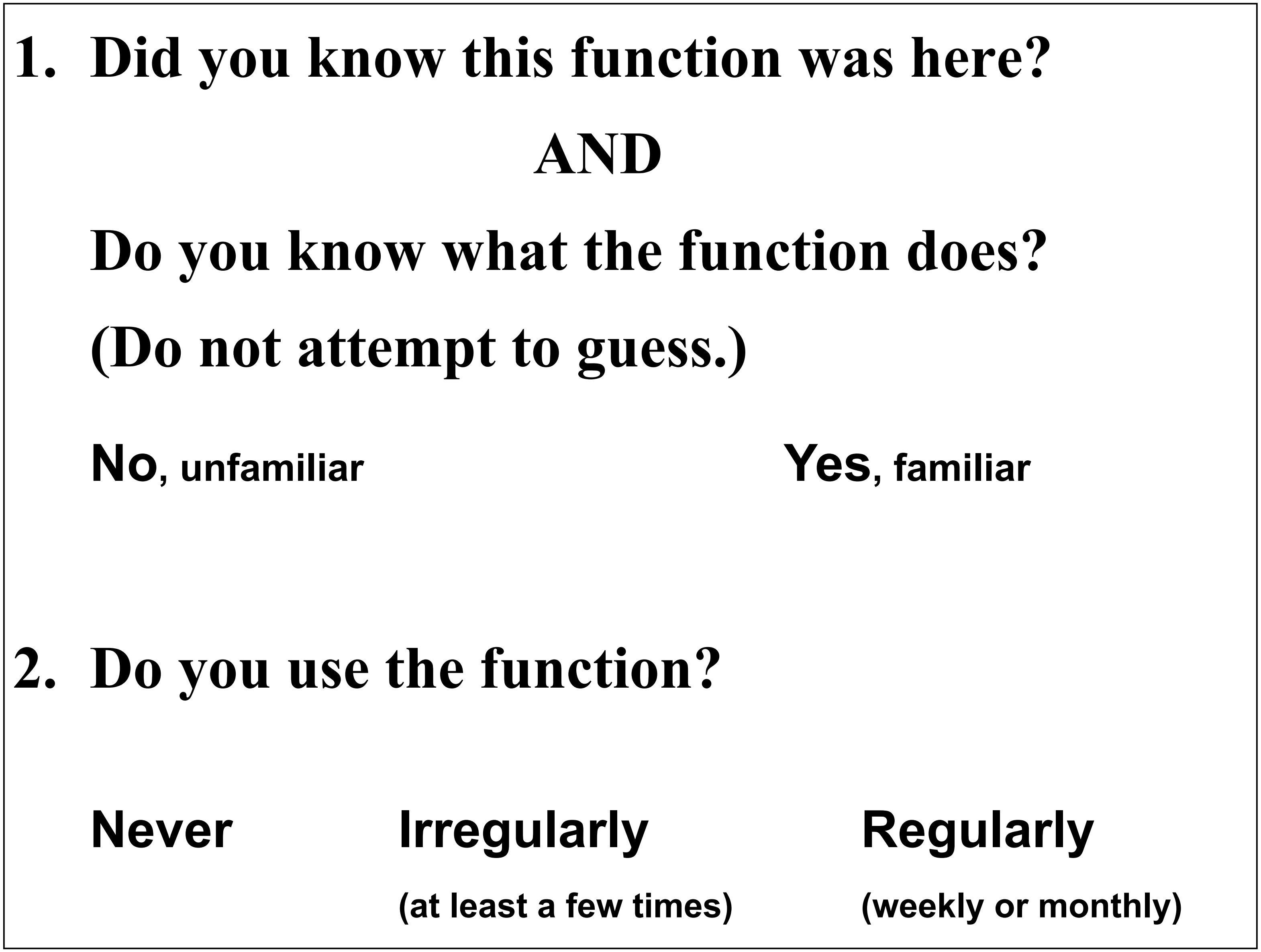
We identified 888 functions in the 71 first-level dialog boxes. But 8 of these dialog boxes are duplicates (or nearly duplicates) of others, which brings the unique function count to 709.

Appendix C: Study One Sample Screen Capture

This sample screen capture shows the **File** menu.



Appendix D: Study One Response Scale



1. Did you know this function was here?

AND

Do you know what the function does?

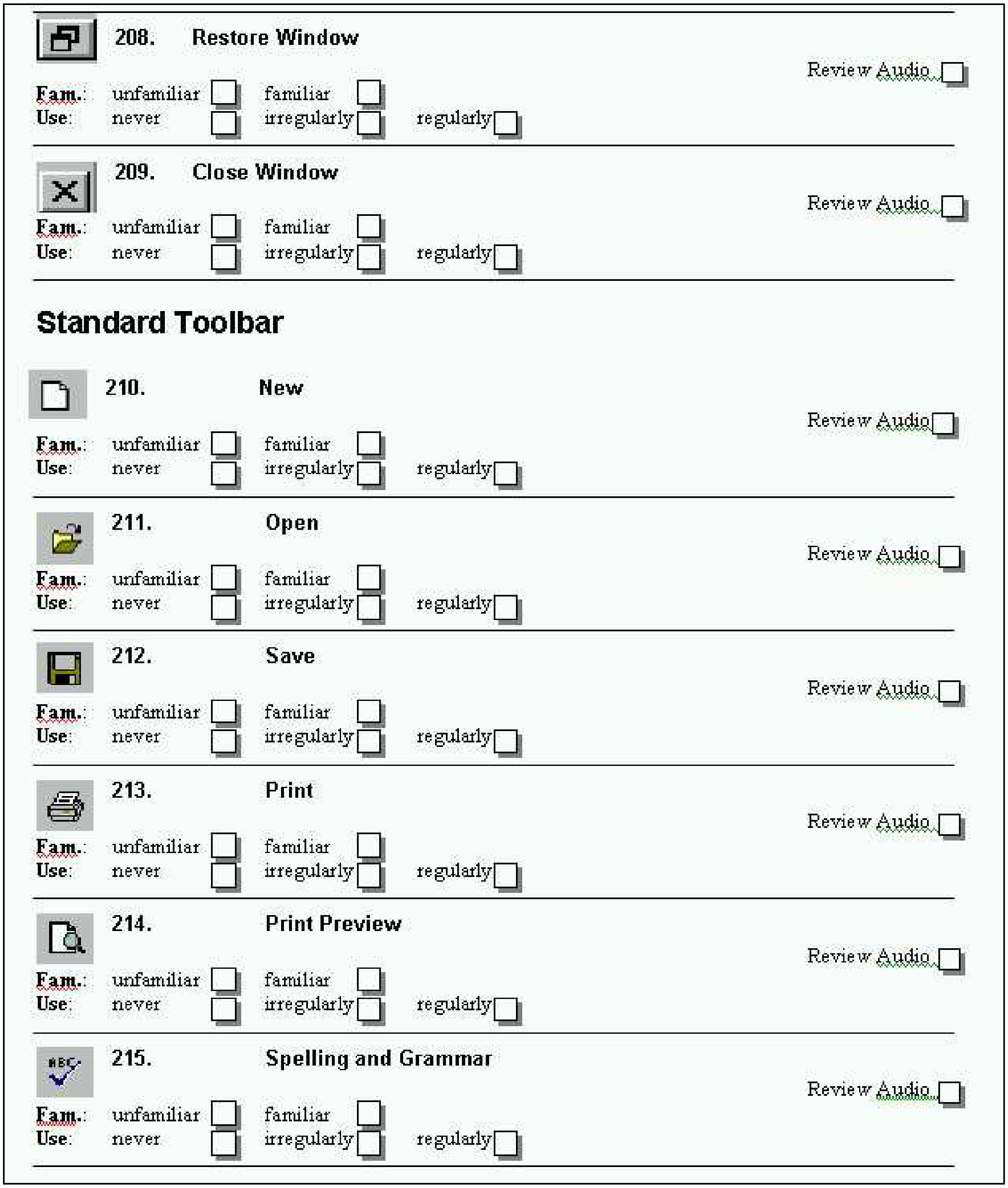
(Do not attempt to guess.)

No, unfamiliar   Yes, familiar

2. Do you use the function?

Never	Irregularly	Regularly
	(at least a few times)	(weekly or monthly)

Appendix E: Study One Functionality Schedule



208. Restore Window

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

209. Close Window

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

Standard Toolbar

210. New

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

211. Open

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

212. Save

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

213. Print

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

214. Print Preview

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

215. Spelling and Grammar

Review Audio □

Fam.: unfamiliar □ familiar □
Use: never □ irregularly □ regularly □

Appendix F: Study One Questionnaire

Experiencing Word Processing

This questionnaire is part of a study to gain insight into users' experiences with word processing. We would appreciate if you would answer the following questions which should not take more than 30 minutes of your time.

Part I: The Production of Texts and Documents

In this section we are interested in your work practices and experiences in writing and publishing texts and documents such as letters, articles, manuscripts etc., regardless of whether you are using a word processor to carry out these activities.

1. **On average, how many hours a day do you spend writing, editing, etc.?**

□ less than 15 minutes
□ less than one hour
□ one to three hours
□ three to five hours
□ more than five hours

2. **Of that, what percent is spent**

Creating content ______%
Editing ______%
Formatting/page layout ______%
Other, please specify: ____________________ ______%

3. **Please respond to the following statements about how you work**

1 = Never
2 = Seldom
3 = Often
4 = Regularly

I work alone	1	2	3	4
I work with publishers/editors	1	2	3	4
I work with colleagues	1	2	3	4
I work with support staff	1	2	3	4
Other, please specify: ____________________	1	2	3	4

4. What do you produce when you write? (This means any writing regardless of whether you use a computer).

1 = Never
2 = Seldom
3 = Often
4 = Regularly

Letters	1	2	3	4
Manuscripts	1	2	3	4
Magazine or newspaper articles	1	2	3	4
Fiction (e.g., novels, short stories)	1	2	3	4
Academic papers (e.g., journal articles, book chapters, course papers)	1	2	3	4
Textbooks	1	2	3	4
Newsletter-style publication	1	2	3	4
Other, please specify: ____________________	1	2	3	4

5. Please indicate how often you perform the following activities:

1 = Never
2 = Seldom
3 = Often
4 = Regularly

I produce camera-ready copy	1	2	3	4
I put information on the World Wide Web	1	2	3	4
I create a table of contents	1	2	3	4
I create an index	1	2	3	4
I create a bibliography	1	2	3	4
I use footnotes	1	2	3	4

6. **Please indicate how familiar you are with the following:**

1 = Not at all familiar
2 = Somewhat familiar
3 = Very familiar

Proofreaders marks	1	2	3
Galleys	1	2	3
Endnotes	1	2	3
Justification	1	2	3
Widows and orphans	1	2	3

7. **How would you characterize yourself in terms of your knowledge about the overall process of writing and publishing texts and/or documents?**

❑ I have basic knowledge.
❑ I have moderate knowledge.
❑ I have extensive knowledge.

8. **Is there any other relevant information about your writing experiences that you would like to note here?**

____

____

____

____

(continued on next page)

Part II Personal History of Computer Use

9. In what year did you first use a computer? _________

10. What kind of computer have you used?
<u>Tick all that apply</u>

- □ I don’t recall
- □ Mac
- □ PC (DOS only)
- □ PC (Windows, NT)
- □ PC (Linux)
- □ Unix Workstation
- □ Mainframe
- □ Other(s), please specify: _____________________

11. On an average work day, approximately how many hours do you spend using a computer?

_____ hours

12. How often do you use the following computer applications?

	Seldom or Never	**Monthly**	**Weekly**	**Daily**
Word Processor (e.g., Word, WordPerfect)	□	□	□	□
E-mail (e.g., Eudora, Pegasus)	□	□	□	□
Web browser (e.g., Netscape, Internet Explorer)	□	□	□	□
Spreadsheet (e.g., Excel, Lotus 1-2-3)	□	□	□	□
Graphics software (e.g., Corel Draw, Adobe)	□	□	□	□
Presentation software (e.g., Powerpoint, Freelance)	□	□	□	□
Database (e.g., Oracle, dbase, Library Master, EndNote)	□	□	□	□
Online Catalogues or Databases (not available through a web browser)	□	□	□	□
Computer games	□	□	□	□
Other, please specify: _______________________	□	□	□	□

13. Who pays to keep the computer *that you use most often* up-to-date?

- □ I do
- □ My company/institution does

14. Have you written a computer program?

- □ No
- □ Yes

If yes, which statement best describes your programming experience?

- □ I have written less than 1,000 lines of code.
- □ I have written between 1,000 and 5,000 lines of code.
- □ I have written more that 5,000 lines of code.

15. Which of the following have you done with a computer?
<u>Tick all that apply</u>

- □ I have installed an application
- □ I have installed an operating system
- □ I have formatted a hard drive
- □ I have added a new external device (e.g., printer, scanner, external modem)
- □ I have added memory
- □ I have added a new internal device (e.g., diskette drive, hard drive, CD ROM, internal modem)

16. If you were asked to explain how your computer works , which of the following statements best describes your ability to do so:

- □ I have detailed technical knowledge and can explain the internal workings in reasonable detail (e.g., not only can I identify the above mentioned components but I can explain how they are integrated and how software "runs" on hardware).
- □ I have a general working knowledge and can explain the basics (e.g., I can identify the main components including the mother board, the hard drive, the disk drives, the micro processor, RAM etc. and I can explain the difference between hardware, application software, and the operating system).
- □ I can't explain how it works.

17. How would you characterize yourself in terms of your knowledge about computers?

- ❏ I have basic knowledge.
- ❏ I have moderate knowledge.
- ❏ I have extensive knowledge.

18. Is there any other relevant information about your use of computers that you would like to note here?

____

____

____

____

(continued on next page)

Part III: Personal History of Word Processing

19. Do you recall the name of the first word processor that you used and the year that you began using it if you recall?

❑ No, I don't recall

❑ Yes, I used:

- ❑ Microsoft Word Year: ______
- ❑ Word Perfect Year: ______
- ❑ Word Star Year: ______
- ❑ Other, please specify: _____________ Year: ______

20. Do you recall the type of computer you were using at that time?

❑ No, I don't recall

❑ Yes, I was using a:

- ❑ Mac
- ❑ PC (DOS)
- ❑ PC (Windows, NT)
- ❑ PC (Linux)
- ❑ Unix Workstation
- ❑ Mainframe
- ❑ Other(s), please specify: ________________________

21. Please indicate which of the following word processors you have used; the versions of each package that you recall using; and the number of years that you used each version?

❑ Microsoft Word ________________Version(s) _______# of Years

❑ Word Perfect ________________Version(s) _______# of Years

❑ Word Star ________________Version(s) _______# of Years

❑ Other, please specify: ______________________Version(s) _______# of Years

22. How did you first learn word processing?
Tick all that apply:

❑ I took a course.
❑ I used the help feature in the word processor.
❑ I learned informally on my own through trial and error.
❑ I learned informally on my own by reading the manuals and other documentation.
❑ I learned informally from a colleague or friend.
❑ I used commercially available books (sometimes called 3rd party manuals).
❑ Other, please specify: __________________________

Please tell us how useful each of the items you ticked were and why.

____

____

____

____

23. How have you continued to learn about word processing since?
Tick all that apply:

❑ I have taken a course(s).
❑ I used the help feature in the word processor.
❑ I have learned informally on my own through trial and error.
❑ I have learned informally on my own by reading the manuals and other documentation.
❑ I have learned informally from a colleague or friend.
❑ I used commercially available books (sometimes called 3rd party manuals).
❑ Other, please specify: __________________________

Please tell us how useful each of the items you ticked were and why.

____

____

____

____

Part IV: Current Word Processing Practices

24. In an average work day, approximately how many hours do you spend word processing?

□ less than 15 minutes
□ less than one hour
□ one to three hours
□ three to five hours
□ more than five hours

25. In addition to using MS Word do you currently use another word processor?

□ No
□ Yes

If yes, please specify the name(s) of the other word processor(s) and the percentage of your word processing time that you spend using each of them:

MS Word: _______ %
Other(s), please specify ___________________ _______ %

The remainder of the questions in this section refer to MS Word.

26. How do you execute the commands in MS Word that you use <u>regularly</u> (e.g., print, save, open, cut and paste)?
<u>Tick all that apply</u>

Command	**Select from a pull-down menu**	**Click on a toolbar icon**	**Use a hotkey**
Print a single copy	□	□	□
Save	□	□	□
Open	□	□	□
Cut	□	□	□
Copy	□	□	□
Paste	□	□	□
Center text	□	□	□
Bold text	□	□	□

27. When you are trying to do something with MS Word that you haven't done before (e.g., insert a diagram, number a list) how often do you use the following strategies?

1 = Never
2 = Seldom
3 = Often
4 = Regularly

I use help from the pulldown menu in MSWord	1	2	3	4
I use MSWord's Office Assistant (paperclip)	1	2	3	4
I ask a colleague or a friend	1	2	3	4
I use the manual	1	2	3	4
I keep trying different things	1	2	3	4
I try to find a way around it	1	2	3	4
I use commercially available books	1	2	3	4
I give up	1	2	3	4
Other, please specify: _________________	1	2	3	4

28. Please indicate how often you do the following activities in MS Word.
For those activities with which you are unfamiliar please tick the first box rather than using the scale

1 = Never
2 = Seldom
3 = Often
4 = Regularly

	Unfamiliar				
I edit my documents on the computer screen.	□	1	2	3	4
I edit/create styles.	□	1	2	3	4
I change the tab settings.	□	1	2	3	4
I use the "Track Changes" feature.	□	1	2	3	4
I change the page margins.	□	1	2	3	4
I add headers and footers.	□	1	2	3	4
I change the line spacing and the amount of spacing before and after paragraphs.	□	1	2	3	4

Question 28 continued…

1 = Never
2 = Seldom
3 = Often
4 = Regularly

	Unfamiliar				
I edit the look of bulleted and numbered lists.	❑	1	2	3	4
I create columns.	❑	1	2	3	4
I change the width of the columns.	❑	1	2	3	4
I create tables.	❑	1	2	3	4
I change the look of the gridlines in the tables.	❑	1	2	3	4
I create my own macros.	❑	1	2	3	4
I create graphics (figure, logo, image, etc.).	❑	1	2	3	4
I anchor and lock graphics to paragraphs.	❑	1	2	3	4
I insert electronic cross-references.	❑	1	2	3	4
I use the "Revisions" feature.	❑	1	2	3	4
I set up automatically-generated table of contents.	❑	1	2	3	4

29. How would you characterize yourself in terms of your overall knowledge of MS Word?

❑ I have basic knowledge.
❑ I have moderate knowledge.
❑ I have extensive knowledge.

30. Is there any other relevant information about your word processing practices that you would like to note here?

____

____

____

____

Part V: Personal Experience of Word Processing

Thinking about the computer you use most often please answer the following questions.

31. When you are using MS Word, do you have to wait for your computer to complete tasks? *Select the statement that best matches your experience.*

□ Yes, the computer responds slowly to most of my commands.

□ Sometimes -the computer responds slowly.

□ No, it is rare that I have to wait for the computer to respond.

32. Have you ever upgraded your MS Word software? (In other words, have you had a new version of software installed?)

□ No
□ Yes ↓

If yes:

Which statement best describes your attitude towards upgrading the software? *<u>Select one</u>*

□ I resent having to upgrade, but I feel forced to upgrade by the company/institution that I work for.
□ I resent having to upgrade, but I feel forced to upgrade by the software industry.
□ I have no opinion one way or the other about upgrading.
□ Upgrading is a necessary evil, but the gain outweighs the pain.
□ Upgrading is an opportunity, I am eager to have the most up-to-date software.

Which statement(s) describes your reason for upgrading your software? *<u>Tick all that apply</u>*

□ I share documents with others and so my software has to be at a similar level.
□ There are features in the new version of the software that I needed.
□ I did not make the decision to upgrade; it was made on my behalf.
□ No particular reason, I just did it.
□ I upgrade in order to keep up with the latest technology.

33. Given your current computer setup, please indicate the extent to which you agree or disagree with the following statements:

1 = Strongly Disagree
2 = Disagree
3 = No opinion
4 = Agree
5 = Strongly Agree

I feel that it takes me a long time to learn or become comfortable with new versions of MS Word	1	2	3	4	5
I use most of the functions in MS Word.	1	2	3	4	5
I can do my work more *efficiently* (i.e., faster*)* with each new version of MS Word.	1	2	3	4	5
I can do my work more *effectively* (i.e., better) with each new version of MS Word.	1	2	3	4	5
I am overwhelmed by how much new “stuff” there seems to be with each version of MS Word.	1	2	3	4	5
I have a hard time finding the functions I need unless I use them regularly.	1	2	3	4	5
I feel that the time I spend learning a new version of MS Word makes me more *efficient* (i.e., faster).	1	2	3	4	5
I feel that the time I spend learning a new version of MS Word makes me more *effective* (i.e., better) in doing my work.	1	2	3	4	5
Each new version of MS Word becomes easier to use.	1	2	3	4	5
I get annoyed when I can’t quickly find a function that I’ve used previously.	1	2	3	4	5
I feel that the time I spend learning a new version of MS Word makes it less frustrating to use.	1	2	3	4	5

Question 33 continued…

1 = Strongly Disagree
2 = Disagree
3 = No opinion
4 = Agree
5 = Strongly Agree

Each new version of MS Word is more complex than the previous version.	1	2	3	4	5
I resent the time I spend learning a new version of MS Word.	1	2	3	4	5
I get annoyed when I can't quickly figure out how to do something that I need to do.	1	2	3	4	5
Even though there is lots of "stuff" in each new version that I don't use, I'm sure it is there for a good reason.	1	2	3	4	5
After using a new version for a short time, the commands and icons that I don't use don't get in my way	1	2	3	4	5
I think that MS Word is designed for users with more sophisticated needs than myself.	1	2	3	4	5
I feel confident that I will be able to learn what I need to know when I get a new version of MS Word.	1	2	3	4	5
In general, I think MS Word gets better with each new version.	1	2	3	4	5

34. Given your current computer setup, please indicate the extent to which you agree or disagree with the following statements:

1 = Strongly Disagree
2 = Disagree
3 = No opinion
4 = Agree
5 = Strongly Agree

Even though there are functions found in the menus/toolbars that I do not use, it is reassuring to me that they are there.	1	2	3	4	5
If it were up to me I would upgrade MS Word only when new functions that I need are added.	1	2	3	4	5
I would prefer to have only the functions I use in the menus/toolbars – the other functions could be "tucked" away in the event that I might need them at some point.	1	2	3	4	5
I want the latest version of MS Word as soon as it is available.	1	2	3	4	5
I am happy to pay for an application that is "fully loaded" even if I don't use all the functions.	1	2	3	4	5
I would like a word processor that gave me only the functions that I use.	1	2	3	4	5
It is important for me that my word processor is state-of-the-art.	1	2	3	4	5
I prefer to continue using the version of MS Word currently on my machine for as long as possible.	1	2	3	4	5
It is important to me that I continually discover new functions.	1	2	3	4	5
Wading through unfamiliar functions can often be annoying/frustrating.	1	2	3	4	5

35. Please indicate the extent to which you agree or disagree with the following statements about how you initially react when something goes wrong or something unexpected happens when you are using MS Word.

1 = Strongly Disagree
2 = Disagree
3 = No opinion
4 = Agree
5 = Strongly Agree

I blame myself.	1	2	3	4	5
I assume that the software has been designed badly.	1	2	3	4	5
I assume that that I need more training.	1	2	3	4	5
I wish I understood more about how the word processor actually works.	1	2	3	4	5
I blame the software company.	1	2	3	4	5

(continued on next page)

Part VI: Personal Information

36. In what age group are you?

□ 20-29
□ 30-39
□ 40-49
□ 50-59
□ 60+

37. Gender:

□ Male
□ Female

38. In terms of my current occupation, how would you characterize yourself? *Select one*

□ Writer
□ Administrative assistant
□ Journalist
□ Secretary
□ Academic
□ Professional
□ Technical expert
□ Student
□ Designer
□ Administrator/Manager
□ Other, please specify: ________________________________

Please specify area/discipline/field:

____

39. Education:

Please specify degrees/programmes:

□ Some high school
□ Completed high school
□ Some post-secondary education
□ Completed community college ________________________________
□ Completed undergraduate degree ________________________________
□ Some graduate or professional school________________________________
□ Completed postgraduate degree ________________________________

If you could change/improve something in MS Word, what would it be?

Please feel free to add some general comments on your experiences with word processing.

Thank you for your cooperation.
Your time and effort are appreciated.

Appendix G: Study One Feature Profile Scale

This appendix gives the detailed construction of the Feature Profile Scale (FPS). This scale is a composite of three subscales that relate to the user's perception of the number of functions, the need for complete software (not a light version), and the need for up to date software. We first show the statements used to construct each of these three subscales. These statements are a subset of those found in questions 33 and 34 in our Study One Questionnaire (Appendix F). Participants ranked each of the statements on the following 5-point Likert scale (numerical values shown in parentheses):

Strongly Disagree (1)
Disagree (2)
No Opinion (3)
Agree (4)
Strongly Agree (5)

A reliability analysis was used to create each subscale. Cronbach's alpha (α) was used as the reliability measure and the alpha values are shown below in parentheses.

The statements that begin with [NOT] are those for which the participants' rankings had to be transposed – this had the same effect as making all of the statements oriented positively. (A ranking of Strongly Disagree became Strongly Agree, Disagree became Agree, and No Opinion did not change).

We provide a single unifying statement for each of the subscales and the variable name that is used in the FPS construction. A participant's value for each subscale variable is simply the addition of that participant's rankings for each of the statements in the subscale.

Number of functions (3 statements) (α = .83)

1) [NOT] I have a hard time finding the functions I need unless I use them regularly.
2) [NOT] I get annoyed when I can't quickly find a function that I've used previously.
3) [NOT] I get annoyed when I can't quickly figure out how to do something that I need to do.

Statement: "The number of functions in the in interface does not make it difficult for me to find the function I am looking for."

Variable: tot_functions

Completeness (6 statements) (α = .76)

4) Even though there are functions found in the menus/toolbars that I do not use, it is reassuring to me that they are there.
5) [NOT] I would prefer to have only the functions I use in the menus/toolbars – the other functions could be "tucked" away in the event that I might need them at some point.

6) I am happy to pay for an application that is "fully loaded" even if I don't use all the functions.
7) [NOT] I would like a word processor that gave me only the functions that I use.
8) It is important to me that I continually discover new functions.
9) [NOT] Wading through unfamiliar functions can often be annoying/frustrating.

Statement: "I want a complete version of MS Word even if I don't use all the functions."

Variable: tot_complete

Up-to-dateness (3 statements) (α = .77)

10) [NOT] If it were up to me I would upgrade MS Word only when new functions that I need are added.
11) I want the latest version of MS Word as soon as it is available.
12) [NOT] I prefer to continue using the version of MS Word currently on my machine for as long as possible.

Statement: "I want my MS Word software to be up to date – I want the latest version."

Variable: tot_dateness

Before combing the three subscales together we did a reliability analysis of all 12 statements in the subscales. These statements were found to "hang together", i.e., they were correlated which means that users tended to respond to the statements across the three areas in the same way ($\alpha = 0.81$). This justified creating the composite scale from the three subscales.

We did not simply combine the three subscales through addition to create the FPS. The subscales included different numbers of statements and therefore addition would have given more weight to the completeness subscale which has twice as many statements as each of the other two subscales. We normalized each of the subscale variables first and then added their normalized values together to create the FPS.

Normalized variables were created to have the range [0 1]. The following was computed for each participant individually:

n_functions = (tot_functions - 3) / (15 - 3)

n_complete = (tot_complete - 6) / (30 - 6)

n_dateness = (tot_dateness - 3) / (15 - 3)

The normalization was computed this way so that 0.5 in the normalized subscale would accurately represent the midpoint in the 5-point response scale. Taking the functions subscale as an example, if a participant ranked each of the three statements with a No Opinion (valued at 3 in the response scale), we would like the value of n_functions for that participant to be 0.5:

$$\text{n\_functions} = (9 - 3)/(15 - 3)$$

$$= 0.5$$

Each participant's rating on the normalized FPS was calculated as follows:

$$\text{n\_fps} = \text{n\_nfunctions} + \text{n\_complete} + \text{n\_dateness}$$

Therefore, n_fps had the range [0 3].

The distribution of the cases across the FPS revealed that while there were concentrations at both ends of the scale there was a substantial group near the center. We divided the participants into three equal-sized groups: the feature-keen, the feature-neutral, and the feature-shy. There were no specific statistical rules to base this division on. This was very preliminary work to see if the profiling results reported from Microsoft could at all be supported.

Appendix H: Pilot Study Prototype Images

The screen captures in this appendix show the prototype evaluated in the Pilot Study.

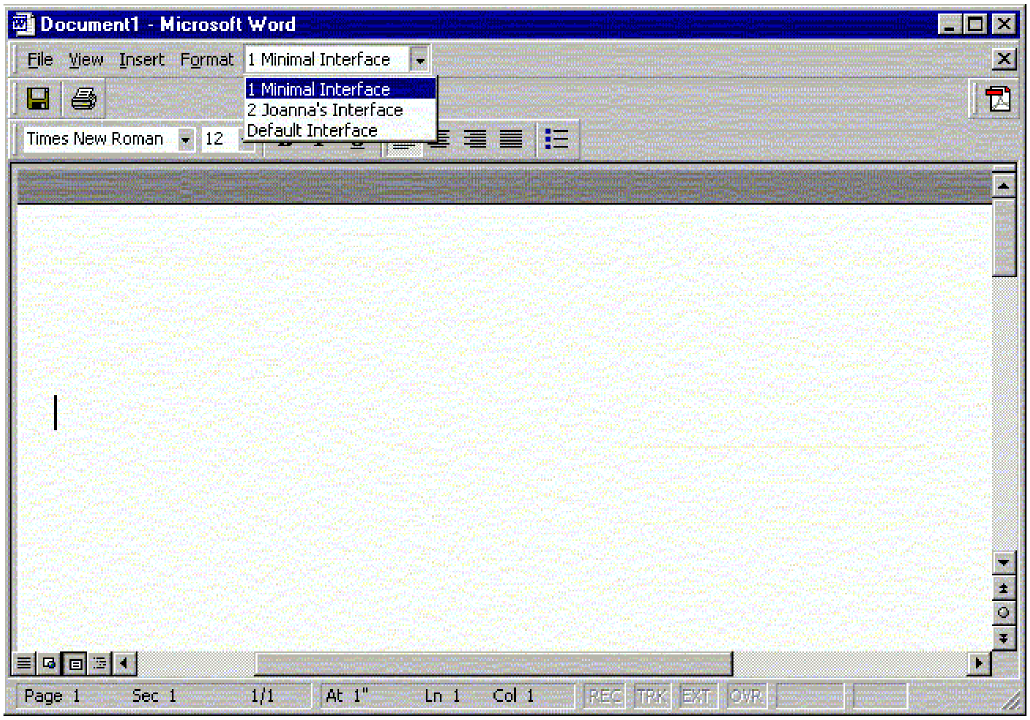


Figure H-1: A toggle on the menu bar allows users to easily switch between the three interfaces. Note that only four out of the nine menus and very few of the toolbar icons are visible in the Minimal Interface, which is the interface shown here, and the interface shown by default when MSWord is opened.

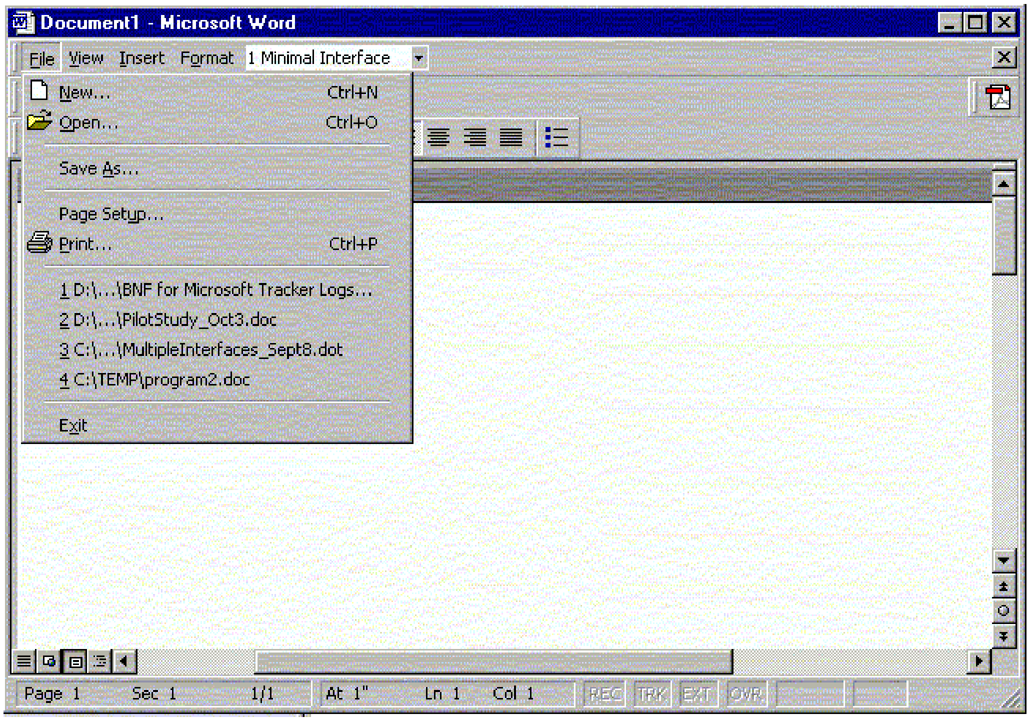


Figure H-2: The **File** *menu in the Minimal Interface has only six menu items plus a list of recently opened documents.*

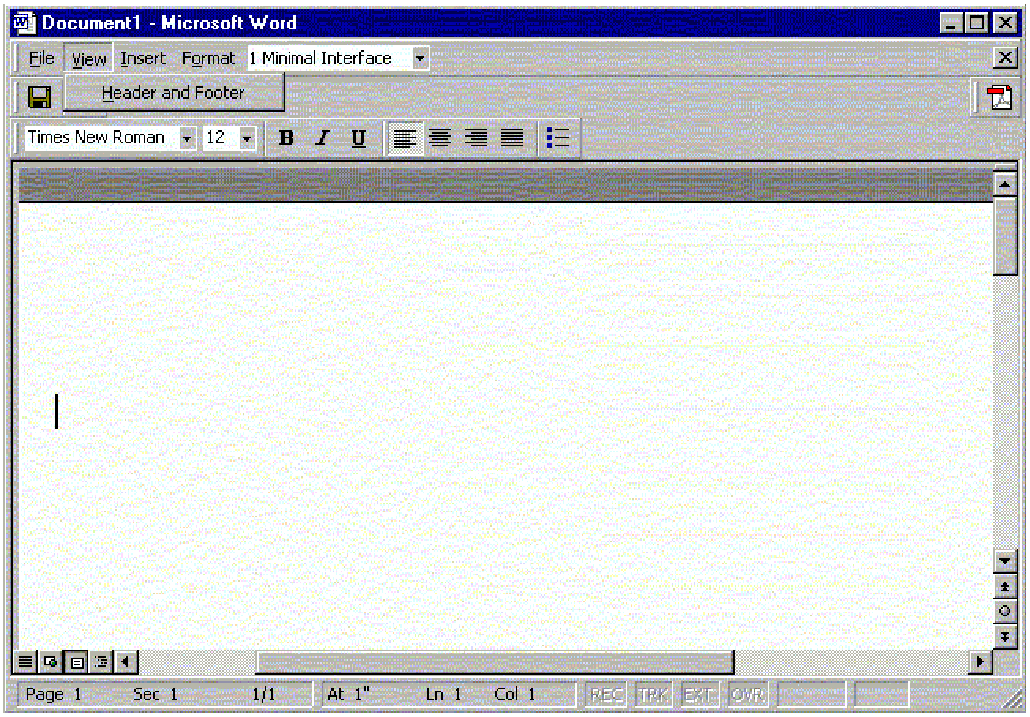


Figure H-3: The **View** *menu in the Minimal Interface has only one menu item.*

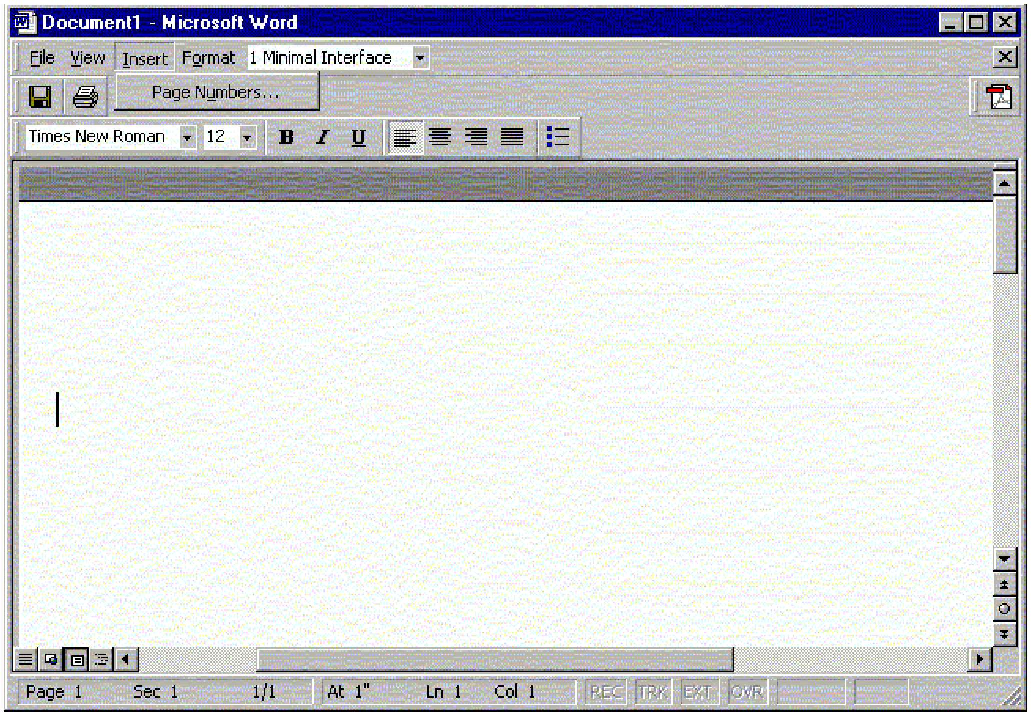


Figure H-4: The **Insert** *menu in the Minimal Interface only provides for page numbering.*

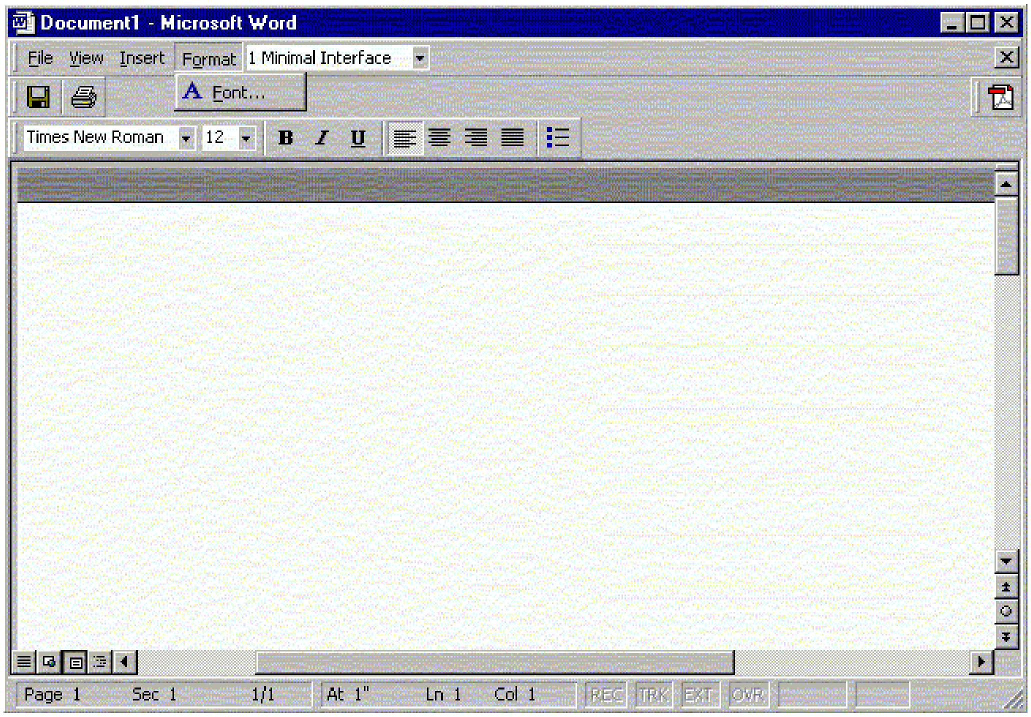


Figure H-5: The **Format** *menu in the Minimal Interface only provides font functionality.*

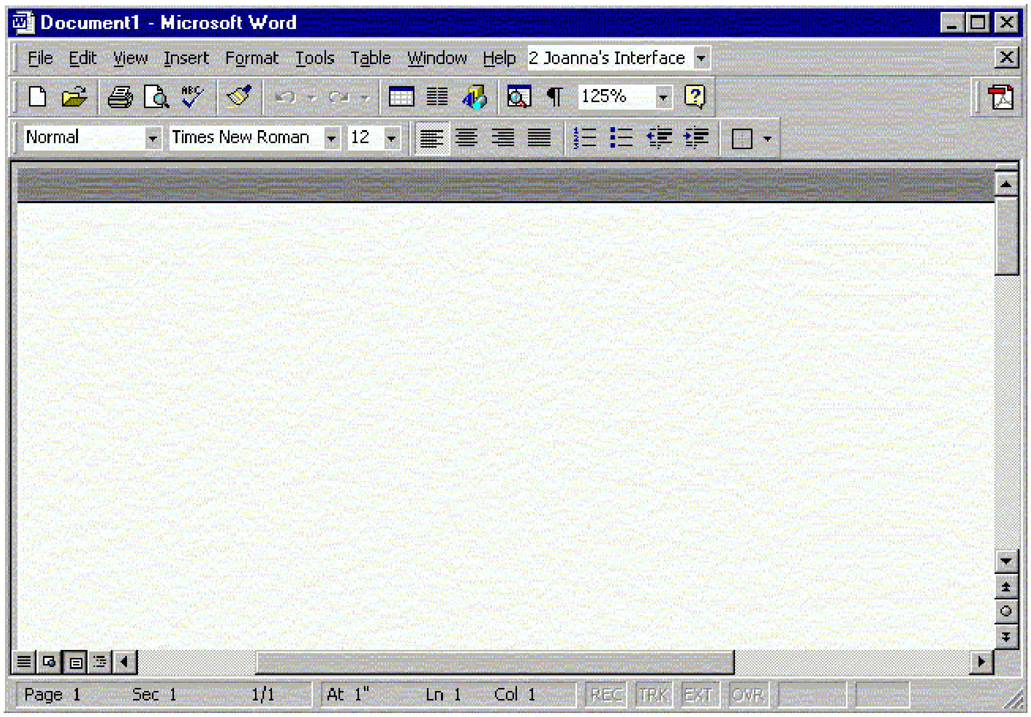


Figure H-6: Joanna's Interface. Note that Joanna's Interface contains a relatively large number of functions for a personalized interface as compared to the personal interfaces of the other 3 participants in the Pilot Study.

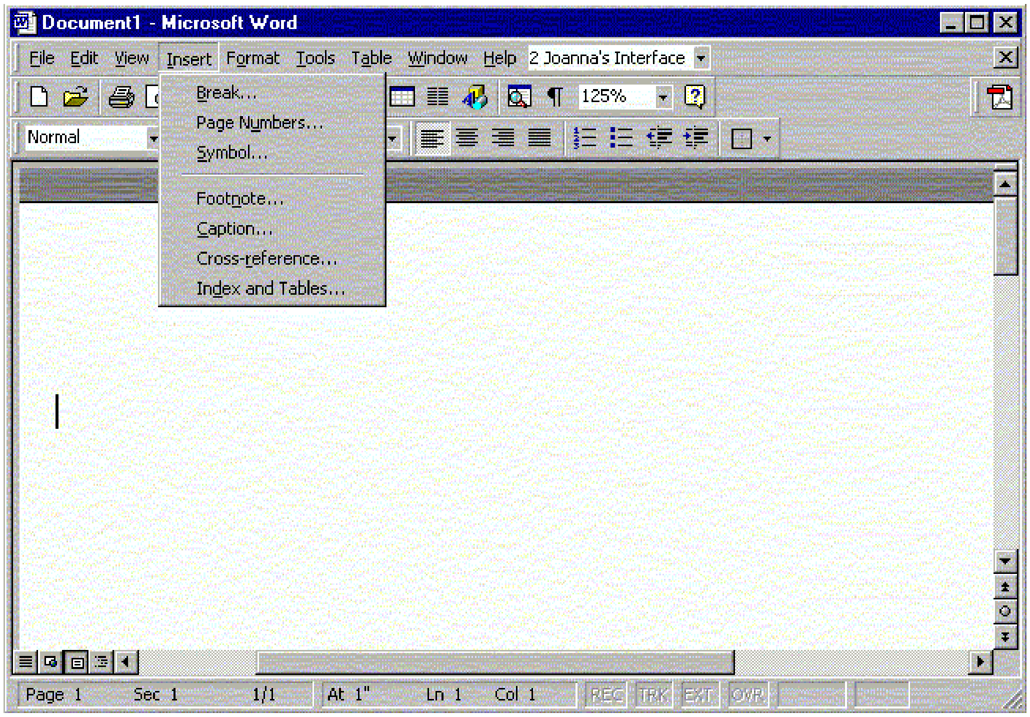


Figure H-7: Joanna's **Insert** *menu contains seven menu items.*

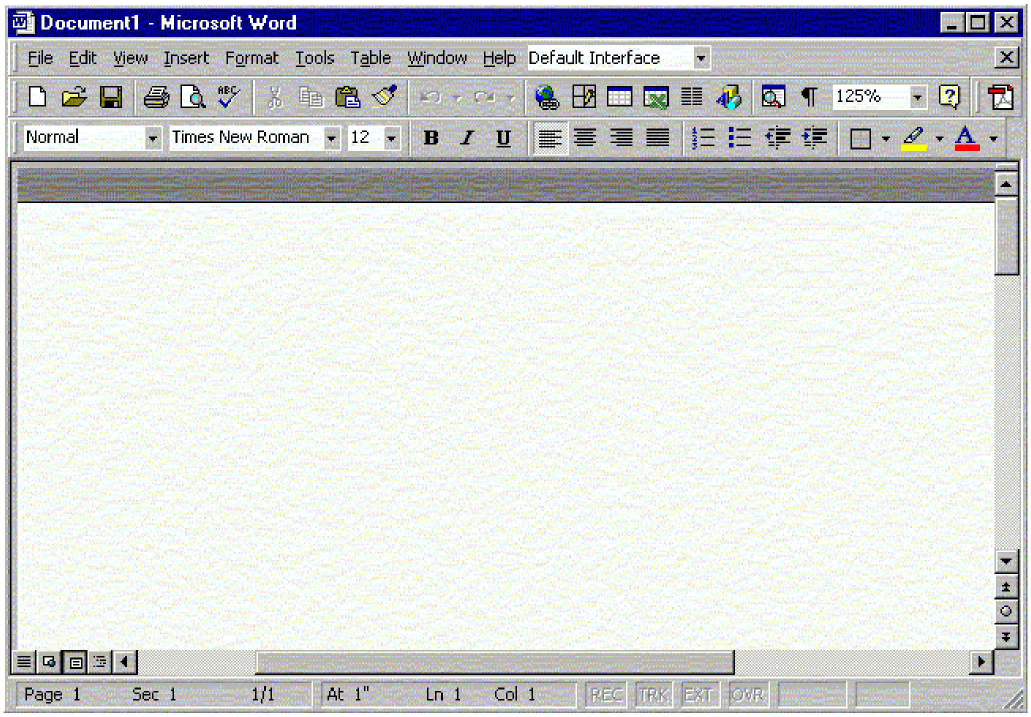

Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help Default Interface
125%
Normal
Times New Roman
12
B I U
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Figure H-8: The Default Interface contains all the functions that would be found in an "out-of-the-box" version of MSWord 2000.

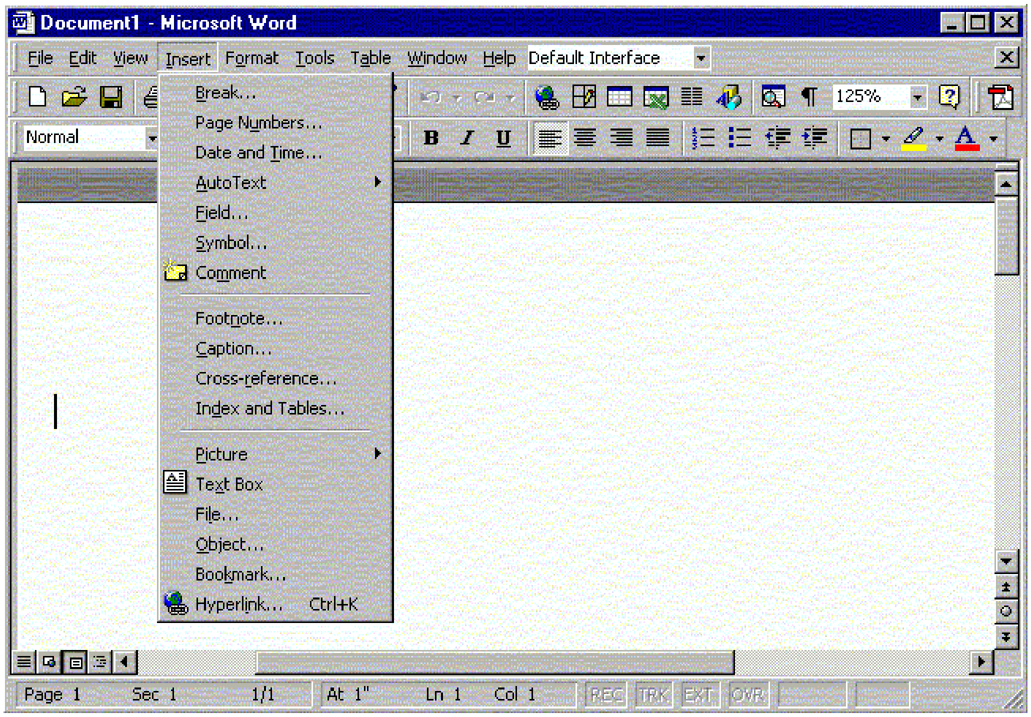
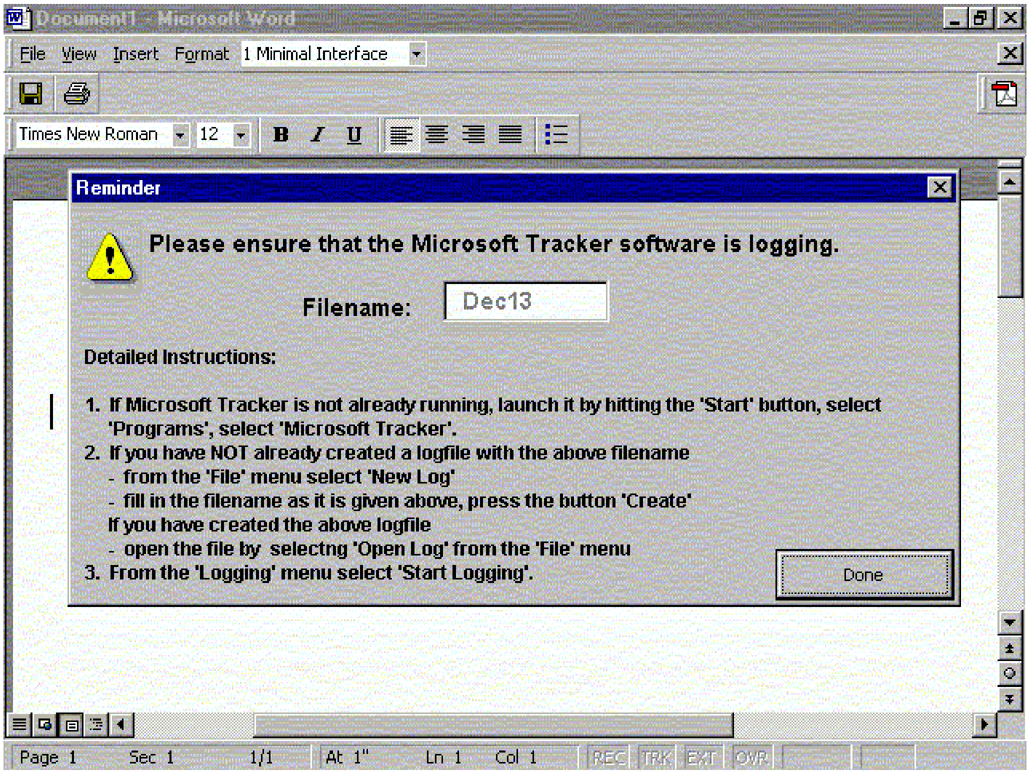


Figure H-9: The **Insert** *menu in the Default Interface contains 17 menu items (plus additional items found in the* **Autotext** *and* **Picture** *submenus, which are not displayed here).*


Document1 - Microsoft Word
File View Insert Format 1 Minimal Interface
Times New Roman 12
B I U
Reminder
Done
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Please ensure that the Microsoft Tracker software is logging.

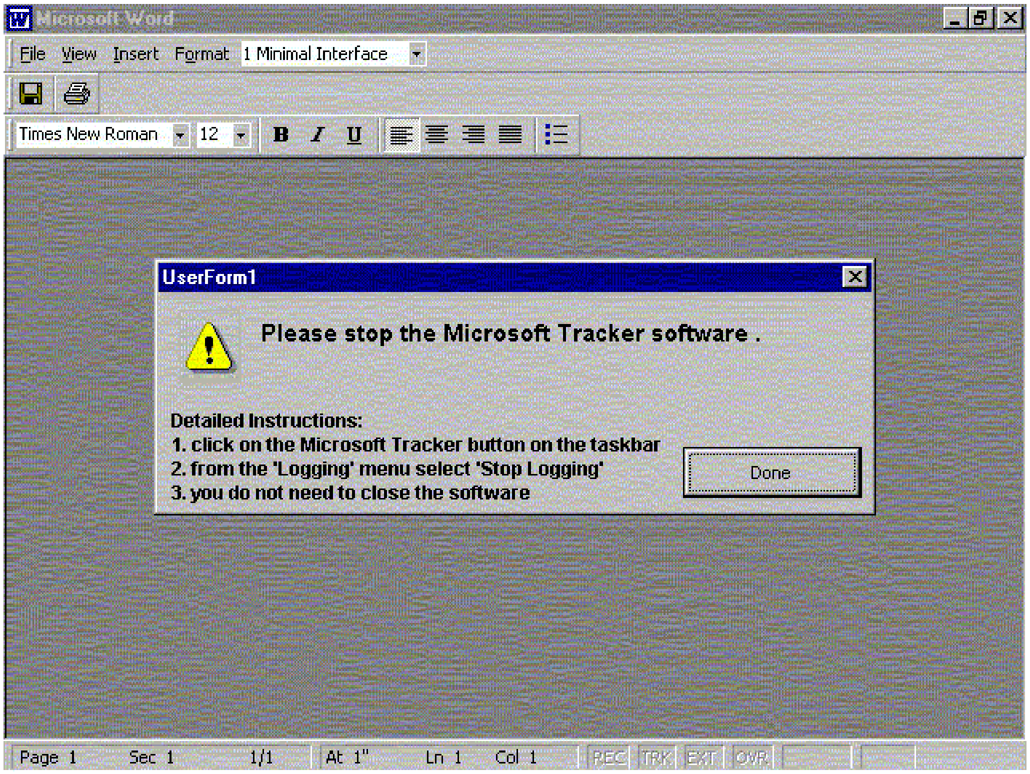
Filename: Dec13

Detailed Instructions:

1. If Microsoft Tracker is not already running, launch it by hitting the 'Start' button, select 'Programs', select 'Microsoft Tracker'.
2. If you have NOT already created a logfile with the above filename
   - from the 'File' menu select 'New Log'
   - fill in the filename as it is given above, press the button 'Create'

   If you have created the above logfile
   - open the file by selectng 'Open Log' from the 'File' menu
3. From the 'Logging' menu select 'Start Logging'.

Figure H-10: A reminder to start the logging software is displayed when MSWord is opened.


Microsoft Word
File View Insert Format 1 Minimal Interface
Times New Roman 12
B I U
UserForm1
Done
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Please stop the Microsoft Tracker software .

Detailed Instructions:
1. click on the Microsoft Tracker button on the taskbar
2. from the 'Logging' menu select 'Stop Logging'
3. you do not need to close the software

Figure H-11: A reminder to stop the logging software is displayed when MSWord is exited.

Appendix I: Study Two Prototype Images

The screen captures in this appendix show the prototype evaluated in Study Two. In particular the process of adding and deleting interface elements is illustrated.

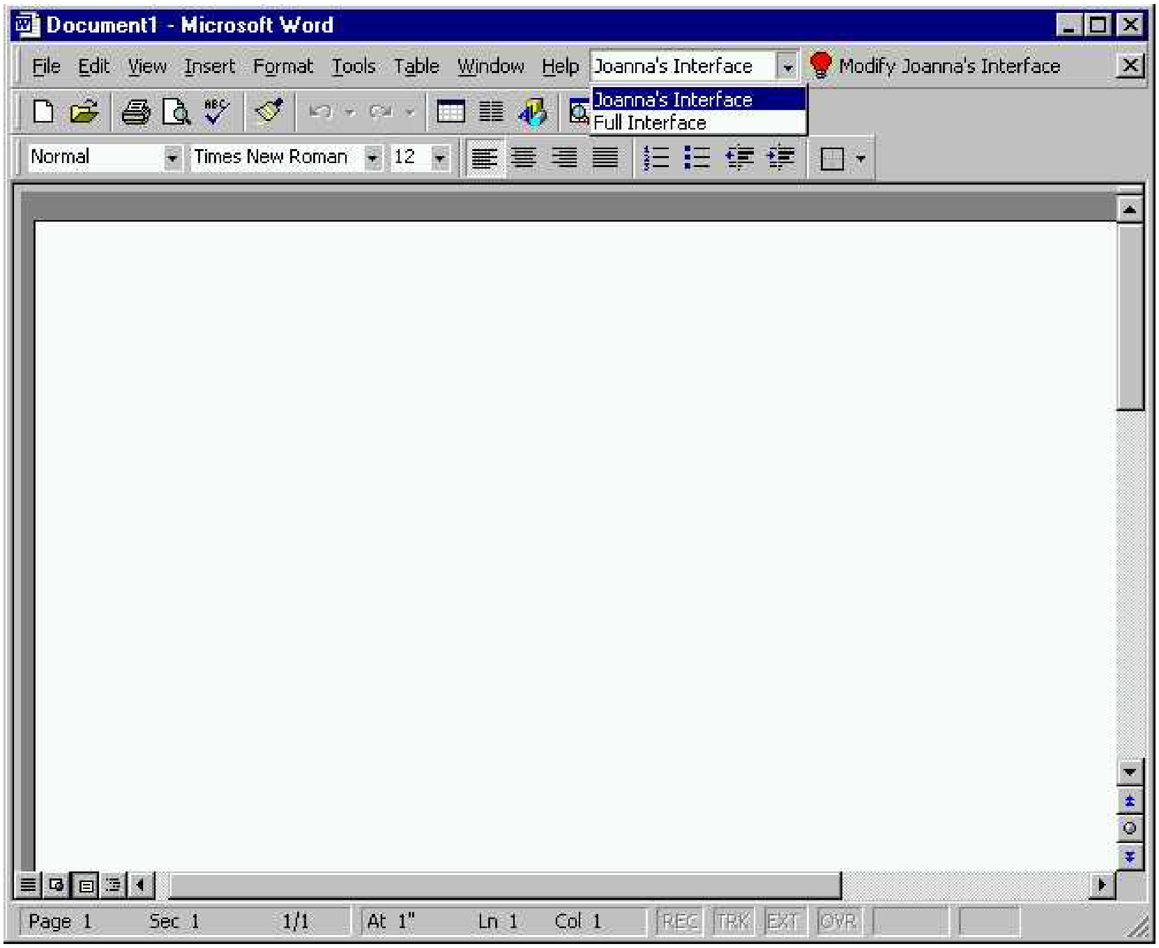
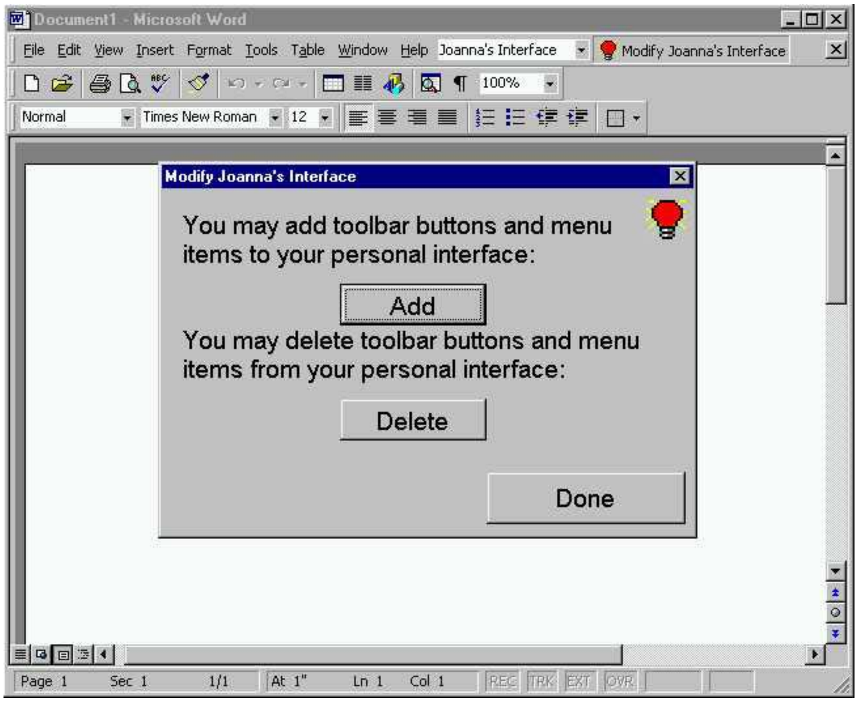


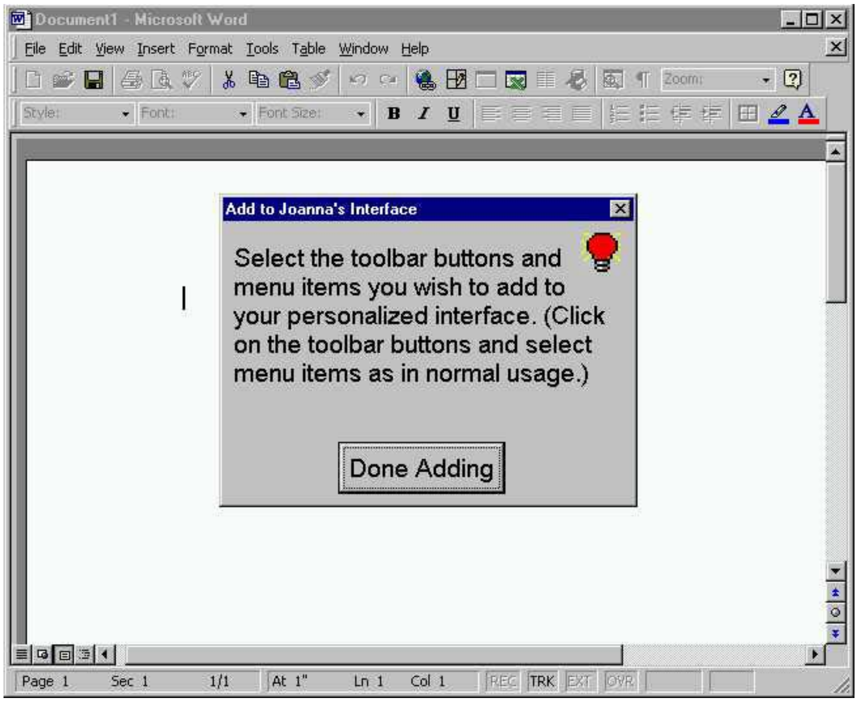
Figure I-1: The same toggle used in the Pilot Study and the new modify button that is labeled **Modify <user name>'s Interface** *to its right. Note that the Minimal Interface has been removed and thus the toggle only contains a personalized interface and the Full Interface (formerly called the Default Interface).*


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Joanna's Interface
Modify Joanna's Interface
Normal
Times New Roman
12
100%
Modify Joanna's Interface
Add
Delete
Done
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

You may add toolbar buttons and menu items to your personal interface:

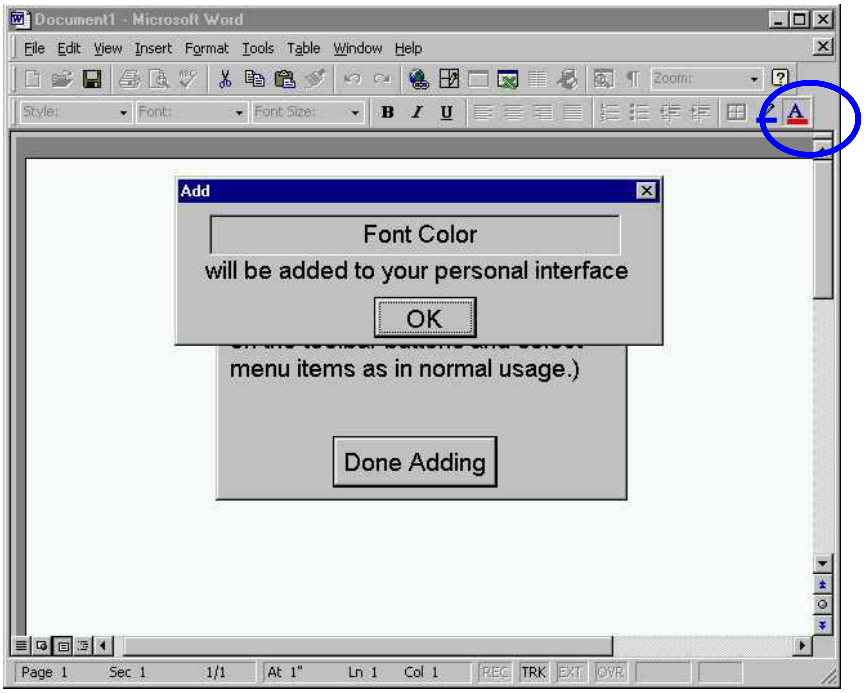
You may delete toolbar buttons and menu items from your personal interface:

Figure I-2: When the **Modify** *button is selected the user is given the option to* **Add** *to or* **Delete** *from his/her personalized interface.*


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Style:
Font:
Font Size:
Zoom:
Add to Joanna's Interface
Done Adding
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Select the toolbar buttons and menu items you wish to add to your personalized interface. (Click on the toolbar buttons and select menu items as in normal usage.)

Figure I-3: When the user chooses to add items to his/her personalized interface, the following dialog is displayed. Note that the user is able to select from toolbars and menus that look virtually identical to those provided in the Full Interface. Those items that are already in the personalized interface are grayed out.

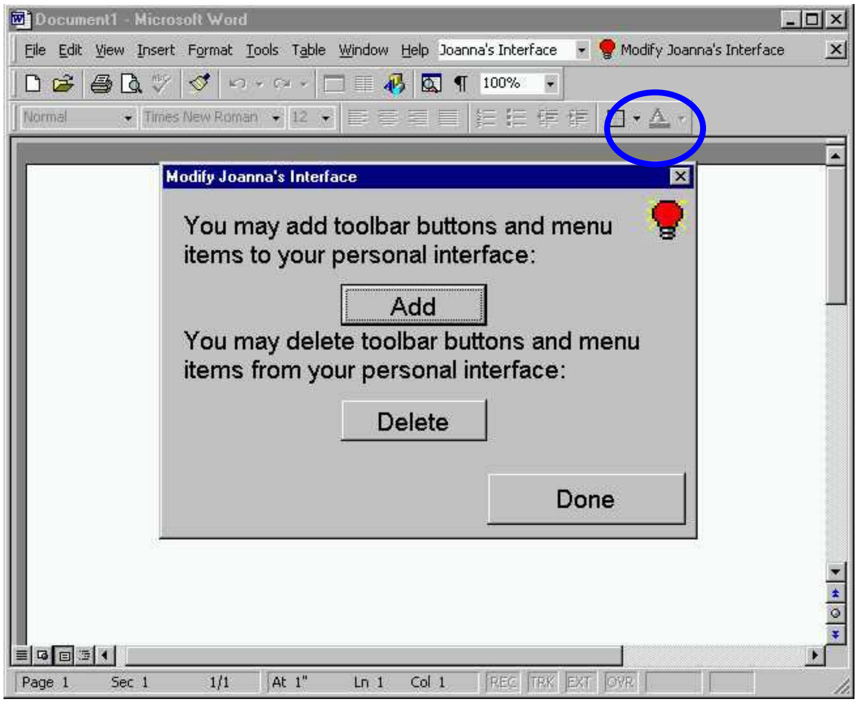

Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Style: Font: Font Size: Zoom:
Add
OK
Done Adding
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Font Color

will be added to your personal interface

menu items as in normal usage.)

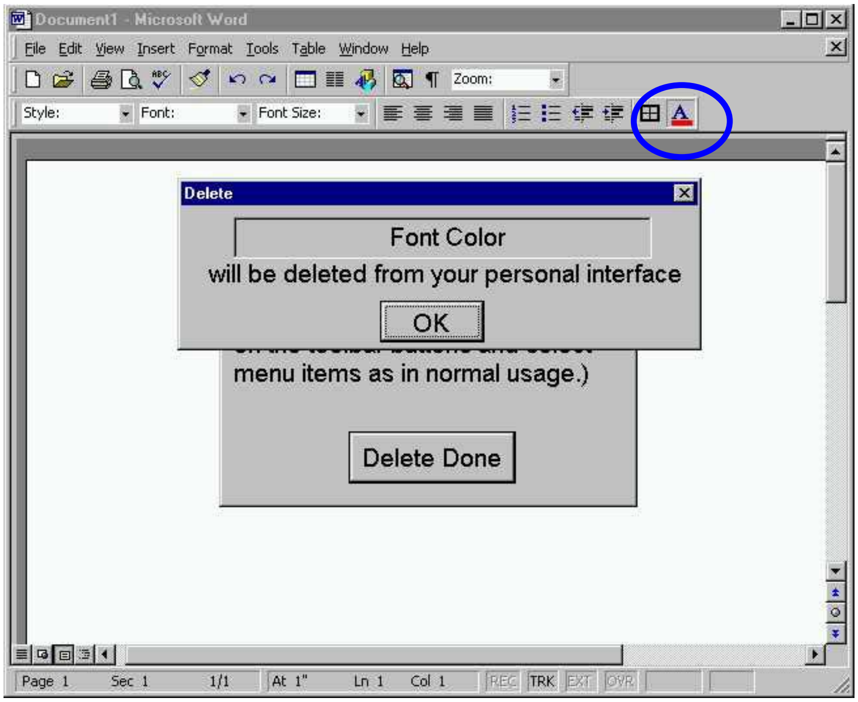
Figure I-4: Here the user has selected the ***Font Colour*** *toolbar item to be added (the circled rightmost item on the* ***Formatting Toolbar****). A dialog box is displayed confirming the selection. At this point the user could continue adding functions by selecting additional toolbar/menu items. When the user has finished adding functions he/she will select* ***Done Adding****.*


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help Joanna's Interface Modify Joanna's Interface
100%
Normal Times New Roman 12
Modify Joanna's Interface
Add
Delete
Done
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

You may add toolbar buttons and menu items to your personal interface:

You may delete toolbar buttons and menu items from your personal interface:

Figure I-5: The user has finished adding a single item and we see that ***Font Colour*** *(circled in image) is now on the user's* ***Formatting Toolbar*** *(which is disabled because no font is being manipulated). The user now has the option to* ***Add*** *more items,* ***Delete*** *items, or finish by selecting* ***Done****.*



Font Color

will be deleted from your personal interface

OK

menu items as in normal usage.)

Delete Done

Figure I-6: The user has chosen to delete items, and is shown here deleting the newly added **Font Colour** *(circled in image). Note that the user is able to select from toolbars and menus that look virtually identical to those provided in his/her Personalized Interface. After confirming, the dialog box shown in Figure I-5 would be re-displayed once the user has completed a sequence of deletions.*

Appendix J: Study Two Call for Participation

Microsoft Word Study Call for Participation

You are invited to participate in an exciting word processing study!

All participants receive a $100 gift certificate for The Bay!

We are looking for participants who:

- are currently using Microsoft Word 2000 and have been using it for one month or more
- use MSWord for 3 hours a week or more
- are 20 years of age or older
- are located close to The University of Toronto downtown campus

What is involved?

Your involvement in this study will last approximately 6 weeks to 2 months. A researcher will come to your place of work on two occasions to make some modifications to the MSWord that runs on your computer. Note that these modifications will not prevent you in any way from carrying out your normal word processing tasks. You will be asked to complete a number of short questionnaires throughout the study on your experiences using MSWord. On the third and final visit the researcher will conclude with a short wrap-up interview. All visits by the researcher will be arranged at a time convenient to you. Apart from using MSWord for your normal word processing tasks, the time required for you to participate in this study (completing questionnaires and meeting with the researcher) is not expected to be more than 3 hours.

To thank you for your participation we will give you a $100 gift certificate for The Bay.

Interested in Participating?

If you are interested in participating we ask that you fill in a preliminary questionnaire:

http://www.dgp.toronto.edu/people/joanna/MSWordStudy.html

This will take no more than 10 minutes of your time. If you have any questions or comments please contact Joanna McGrenere at 416-978-1532 or send email at joanna@dgp.toronto.edu

This research is being supported by:
The Department of Computer Science and
The Knowledge Media Design Institute (KMDI)
University of Toronto

Appendix K: Study Two Preliminary Questionnaire

Preliminary Questionnaire
Microsoft Word Study

Please fill in this questionnaire if you are interested in participating in the Microsoft Word Study. The purpose of this questionnaire is to see if you will be a good fit for the study. Note that all information provided will remain strictly confidential. Completing the questionnaire should require no more than 10 minutes of your time.

**** Note that to participate in this study you must:*

- *currently use Microsoft Word 2000 for at least 3 hours per week (if you are unsure of which version you are using, select "About Microsoft Word" from the Help menu); and*
- *have been using Microsoft Word 2000 for at least one month; and*
- *do the majority of your word processing on a single computer; and*
- *be located centrally (at most a half hour drive away from The University of Toronto -- University and College).*

Please fill in every question as best you can. The fields with [required] are required fields.

Section 0: Contact Information

Firstname: [required]	
Lastname: [required]	
Phone number (including area code): [required]	
E-mail address: [required]	

Section I: Essentials

A. [required] Do you currently use Microsoft Word, version 2000?

○ Yes ○ No

B. [required] For approximately how many months have you used Microsoft Word 2000?

○ 0 - 1 months
○ 1 - 3 months
○ 3 - 6 months
○ more than 6 months

C. [required] On average, how many hours a week do you spend word processing?

○ 0 - 1 hours
○ 1 - 2 hours
○ 2 - 3 hours
○ 3 - 4 hours
○ 4 - 5 hours
○ more than 5 hours

D. [required] Do you do the majority of your word processing on a single computer?

○ Yes ○ No

E. [required] Which operating system are you currently using?

○ Windows 95

○ Windows 98

○ Windows NT

○ Windows 2000

○ Other ______

○ I don't know

○ Note that Microsoft Word 2000 is not available for the Macintosh operating system.

F. [required] Relative to The University of Toronto downtown campus (University and College) which of the following best describes the location at which you carry out your word processing?

○ I am on the downtown campus.

○ I am walking distance from campus (up to 20 minutes walking).

○ I am a short drive from campus (up to 10 minutes drive).
○ I am driving distance from campus (up to 30 minutes drive).
○ I am more than 30 minutes away by car.

Section II: General Experience with MS Word and other MS Office applications.

If you have little or no experience with MS Office applications other than MS Word this is okay. Please continue to fill in the questionnaire.

It is important that you answer the questions in this section as best you can "off the top of your head". Our intention is not to test your ability to look up answers.

[NOTE: Questions 1 through 13 were taken directly from the Microsoft Office screening questionnaire known as Office Knowledge Test (Version 3). This questionnaire assesses level of expertise for the MSOffice product suite and categorizes expertise into five groups: novice, beginner, intermediate, advanced, and expert. It is a proprietary tool and so these questions have been omitted from this dissertation for confidentiality reasons.]

14. Given you current computer setup, please indicate the extent to which you agree or disagree with the following statements[58]:

SD = Strongly Disagree
D = Disagree
N = Neutral
A = Agree
SA = Strongly Agree

I can do my work more *efficiently* (i.e., faster) with each new version of MS Word.	○ SD	○ D	○ N	○ A	○SA

[58] Question 14 includes statements that were taken directly from questions 33 and 34 of our Experiencing Word Processing Questionnaire (Appendix F).

I can do my work more *effectively* (i.e., better) with each new version of MS Word.	○ SD	○ D	○ N	○ A	○SA
I am overwhelmed by how much new "stuff" there seems to be with each version of MS Word.	○ SD	○ D	○ N	○ A	○SA
I have a hard time finding the functions I need unless I use them regularly.	○ SD	○ D	○ N	○ A	○SA
I feel that the time I spend learning a new version of MS Word makes me more *efficient* (i.e., faster).	○ SD	○ D	○ N	○ A	○SA
I feel that the time I spend learning a new version of MS Word makes me more *effective* (i.e., better) in doing my work.	○ SD	○ D	○ N	○ A	○SA
Each new version of MS Word becomes easier to use.	○ SD	○ D	○ N	○ A	○SA
I get annoyed when I can't quickly find a function that I've used previously.	○ SD	○ D	○ N	○ A	○SA
I feel that the time I spend learning a new version of MS Word makes it less frustrating to use.	○ SD	○ D	○ N	○ A	○SA
Each new version of MS Word is more complex than the previous version.	○ SD	○ D	○ N	○ A	○SA
I resent the time I spend learning a new version of MS Word.	○ SD	○ D	○ N	○ A	○SA
I get annoyed when I can't quickly figure out how to do something that I need to do.	○ SD	○ D	○ N	○ A	○SA
Even though there is lots of "stuff" in each new version that I don't use, I'm sure it is there for a good reason.	○ SD	○ D	○ N	○ A	○SA
In general, I think MS Word gets better with each new version.	○ SD	○ D	○ N	○ A	○SA
Even though there are functions found in the menus/toolbars that I do not use, it is reassuring to me that they are there.	○ SD	○ D	○ N	○ A	○SA
If it were up to me I would upgrade MS Word only when new functions that I need are added.	○ SD	○ D	○ N	○ A	○SA
I would prefer to have only the functions I use in the menus/toolbars - the other functions could be "tucked" away in the event that I might need them someday.	○ SD	○ D	○ N	○ A	○SA
I want the latest version of MS Word as soon as it is available.	○ SD	○ D	○ N	○ A	○SA
I am happy to pay for an application that is "fully loaded" even if I don't use all the functions.	○ SD	○ D	○ N	○ A	○SA
I would like a word processor that gave me only the functions that I use.	○ SD	○ D	○ N	○ A	○SA

I prefer to continue using the version of MS Word currently on my machine for as long as possible.	○ SD	○ D	○ N	○ A	○SA
It is important to me that I continually discover new functions.	○ SD	○ D	○ N	○ A	○SA
Wading through unfamiliar functions can often be annoying/frustrating.	○ SD	○ D	○ N	○ A	○SA

Section III: Personal Information [59]

15. In what age group are you?

○ 19 and under
○ 20 - 29
○ 30 - 39
○ 40 - 49
○ 50 - 59
○ 60 +

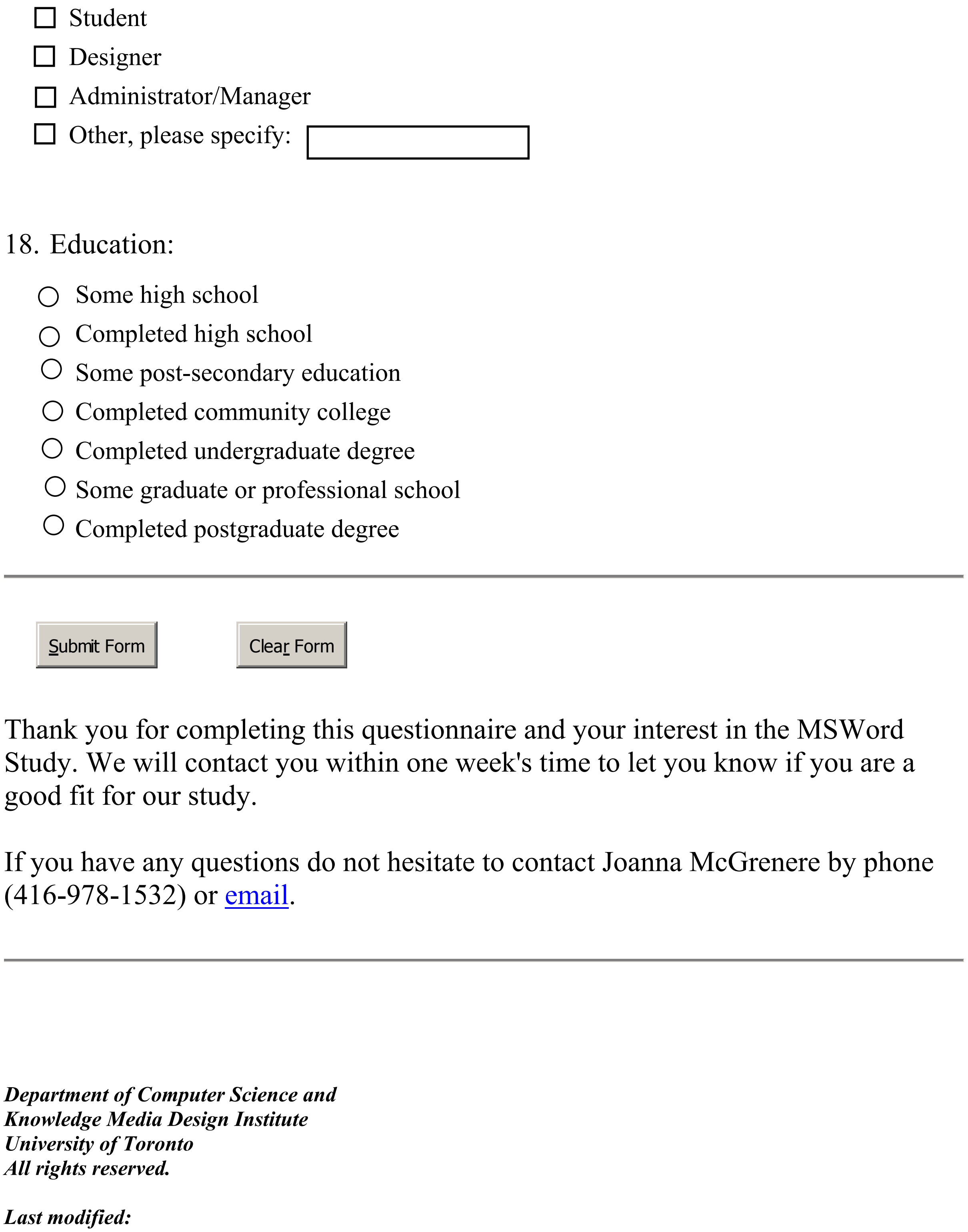
16. Gender:

○ Male
○ Female

17. In terms of your current occupation, how would you characterize yourself?

□ Writer
□ Administrative Assistant
□ Journalist
□ Secretary
□ Academic
□ Professional
□ Technical expert

[59] The questions in Section III of this questionnaire were taken directly from questions 36 through 39 of our Experiencing Word Processing Questionnaire (Appendix F).



□ Student
□ Designer
□ Administrator/Manager
□ Other, please specify: □

18. Education:

○ Some high school
○ Completed high school
○ Some post-secondary education
○ Completed community college
○ Completed undergraduate degree
○ Some graduate or professional school
○ Completed postgraduate degree

Thank you for completing this questionnaire and your interest in the MSWord Study. We will contact you within one week's time to let you know if you are a good fit for our study.

If you have any questions do not hesitate to contact Joanna McGrenere by phone (416-978-1532) or email.

Department of Computer Science and
Knowledge Media Design Institute
University of Toronto
All rights reserved.

Last modified:

Appendix L: Study Two Instructions

First Meeting Instructions

Instructions for Participants - First Meeting

MSWord Personal Installed

A new version of MSWord called MSWord Personal has now been installed on your machine.

MSWord Personal provides you with two interfaces:

1. Your personal interface. Initially this interface has very few functions in its menus and toolbars but you can change it to include whichever functions you want through adding and deleting functions. To add or delete functions use the button labeled "Modify 's Interface" on the main menu bar.

2. The Full Interface. This is the standard interface that contains all of the functions.

You may choose to use these interfaces in any way that you want. For example, you may choose:

1. to use only one of the interfaces (and thereby essentially ignore the other).
2. to use both interfaces, one at a time, switching between them in any way that suits your needs.

It is important to note that there is no right or wrong way to use these interfaces.

If a Problem with MSWord Personal Should Occur

MSWord Personal has been extensively tested and we don't anticipate that you will have any difficulties with this version. MSWord is a reasonably stable software program but it does occasionally have problems (it may at times hang or crash). This version of the software, MSWord Personal, will be no more or less stable than MSWord 2000. We recommend that you continue to use error prevention methods such as saving your work regularly.

If you do experience a problem with this software, please follow the steps below:

1. Restart your system and launch MSWord again.

2. If step (1) does not solve your problem, make a note of the problem you are having - record what you were trying to do when you encountered problems and note any error messages that were displayed. Contact Joanna McGrenere using the contact information provided.

3. If steps (1) and (2) do not solve your problem, you can temporarily uninstall MSWord Personal from your machine. This is very easy to do. Locate the folder C:\MSWordStudy on your hard drive. Double click on the file RemoveMSWordPersonal.bat and follow the instructions provided. (See the README.txt if you are having difficulty.) Once MSWord Personal is uninstalled you will automatically be using MSWord 2000 once again. Please contact Joanna McGrenere immediately if you uninstall MSWord Personal.

Again, we don't expect that anyone will need to uninstall MSWord Personal during this study.

Last modified:

Second Meeting Instructions

Instructions for Participants - Second Meeting

MSWord Personal Uninstalled

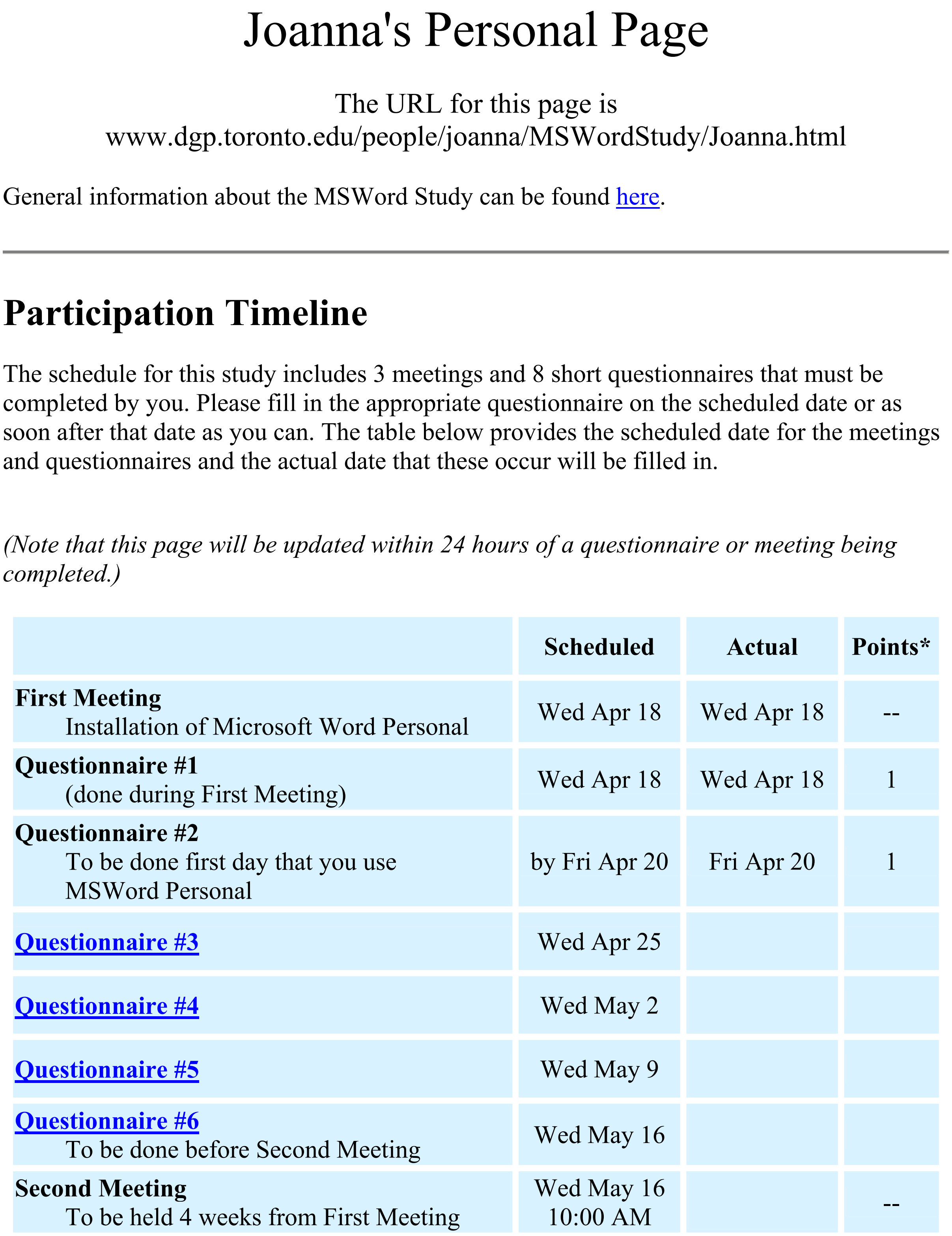
MSWord Personal has now been uninstalled from your computer. You are once again using MSWord 2000 which will appear to you, for the most part[60], the same as it did prior to the installation of MSWord Personal.

The software logger will continue to operate in the background capturing your commands and if you have an Internet connection will continue to send copies of your log files to the UofT site.

Last modified:

[60] The "for the most part" was added here in case there were small differences between the state of the participant's MSWord 2000 interface prior to the Word Personal installation and its state at the point of the Second Meeting. For example, some users had applications external to MSWord such as Adobe Acrobat that automatically add items to the MSWord interface. Some of these additional menu items interacted poorly with the MSWord Personal prototype and, with the participant's permission, had to be disabled when MSWord Personal was installed. They remained disabled until the end of Study Two.

Appendix M: Study Two Personal Web Page



Joanna's Personal Page

The URL for this page is
www.dgp.toronto.edu/people/joanna/MSWordStudy/Joanna.html

General information about the MSWord Study can be found here.

Participation Timeline

The schedule for this study includes 3 meetings and 8 short questionnaires that must be completed by you. Please fill in the appropriate questionnaire on the scheduled date or as soon after that date as you can. The table below provides the scheduled date for the meetings and questionnaires and the actual date that these occur will be filled in.

(Note that this page will be updated within 24 hours of a questionnaire or meeting being completed.)

	Scheduled	Actual	Points*
First Meeting Installation of Microsoft Word Personal	Wed Apr 18	Wed Apr 18	--
Questionnaire #1 (done during First Meeting)	Wed Apr 18	Wed Apr 18	1
Questionnaire #2 To be done first day that you use MSWord Personal	by Fri Apr 20	Fri Apr 20	1
Questionnaire #3	Wed Apr 25		
Questionnaire #4	Wed May 2		
Questionnaire #5	Wed May 9		
Questionnaire #6 To be done before Second Meeting	Wed May 16		
Second Meeting To be held 4 weeks from First Meeting	Wed May 16 10:00 AM		--

Questionnaire #7	Wed May 23		
Questionnaire #8	Wed May 30		
Third (Final) Meeting To be held within a few days of completing Questionnaire #8			--

Points* :
As an incentive to completing the questionnaires exactly on schedule, the organizers of this study are offering a prize to the participant who completes the most number of questionnaires on the correct date. The mechanism for awarding the prize is very simple: there are 8 questionnaires and for each questionnaire completed on schedule there is 1 point awarded. If a questionnaire is completed one day late, half a point is awarded. The participant who has the most points at the end of the study will win the prize. If two or more participants have an equal number of points and these are greater than all other participants, then there will be a draw among those with the most points.

The prize: an *additional* $100 Gift Certificate for The Bay

(Note that this prize is in addition to the gift certificate that all participants receive solely for their participation in the study as outlined in the Microsoft Word Study Consent Form.)

Last modified:

Appendix N: Study Two Q1, Q2, Q7 and Q8

MSWord Study: Questionnaire #1

First name: □
Lastname: □
Date: □

This questionnaire is to be filled in during the First Meeting. *Please fill in every question as best you can.*

History of Word Processing[61]

1. Do you recall the name of the first word processor that you used and the year that you began using it if you recall?

○ No, I don't recall

○ Yes, I used:

○ Microsoft Word Year: □
○ Word Perfect Year: □
○ Word Star Year: □
○ Other: □ Year: □

2. Do you recall the type of computer you were using at that time?

○ No, I don't recall

○ Yes, I was using:

○ Mac
○ PC (DOS)

[61] Questions 1 through 3 of this questionnaire were taken directly from questions 19 through 21 of our Experiencing Word Processing Questionnaire (Appendix F).

○ PC (Windows, NT)

○ PC (Linux)

○ Unix Workstation

○ Mainframe

○ Other:

3. Please indicate which of the following word processors you have used and the number of years that you used each version.

□ Microsoft Word # of Years:

□ Word Perfect # of Years:

□ Word Star # of Years:

□ Other, please specify: # of Years:

Word Processing with Microsoft Word 2000

4. Do you recall the word processor and its version number that you used previous to Microsoft Word 2000 (e.g., Microsoft Word 97, Microsoft Word 6, WordPerfect 8) and approximately how long you used that version?

5. What are the *main differences* between your current version of Microsoft Word (version 2000) and the previous version you used? Please describe these differences as best you can.

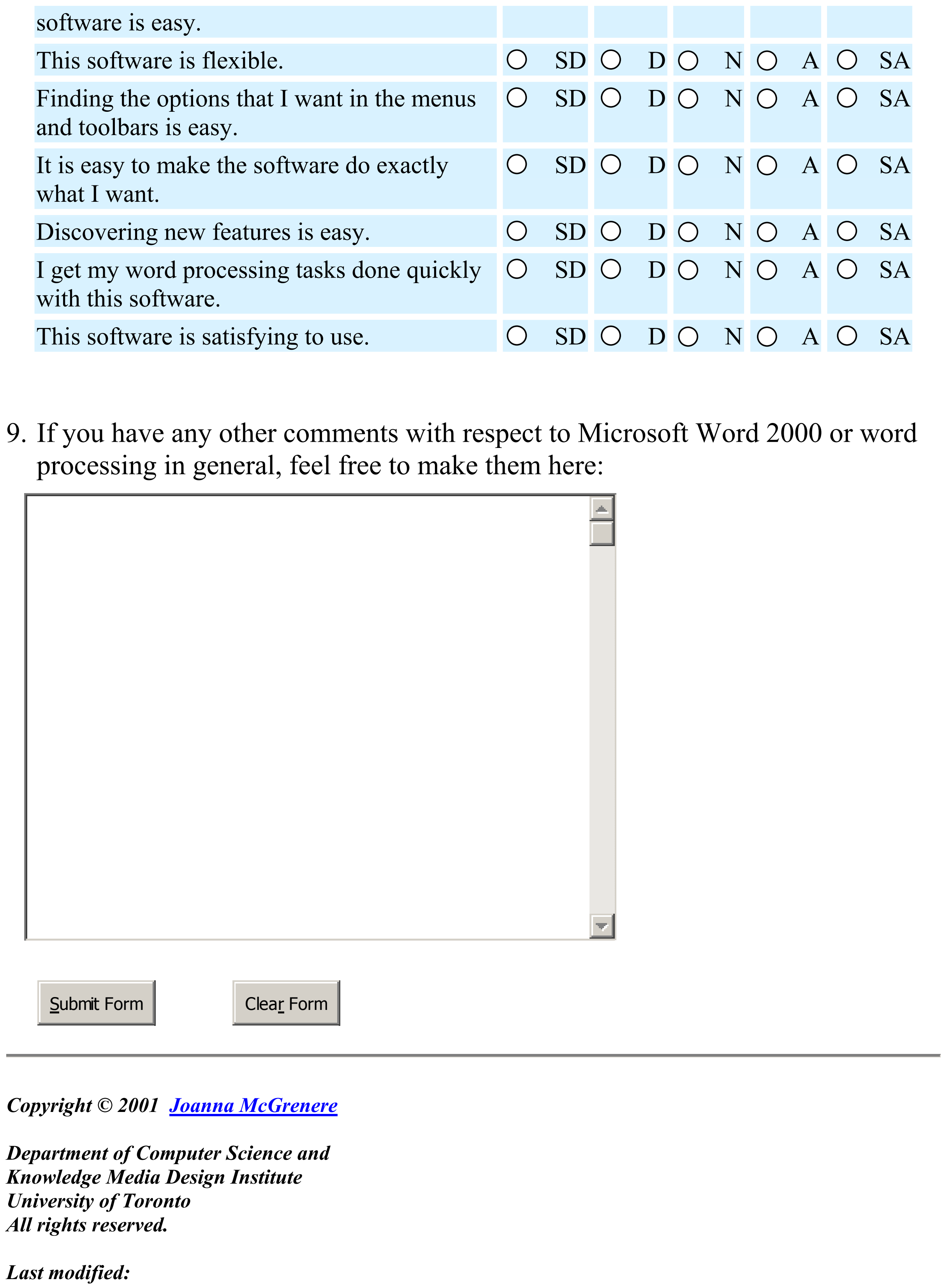
6. What, if any, aspects of your previous version did you prefer to Microsoft Word 2000?

7. What, if any, aspects of Microsoft Word 2000 do you prefer to your previous version?

8. With respect to Microsoft Word 2000, please indicate the extent to which you agree or disagree with the following statements:

SD = Strongly Disagree
D = Disagree
N = Neutral
A = Agree
SA = Strongly Agree

This software is easy to use.	○ SD	○ D	○ N	○ A	○ SA
I am in control of the contents of the menus and toolbars.	○ SD	○ D	○ N	○ A	○ SA
I will be able to learn how to use all that is offered in this software.	○ SD	○ D	○ N	○ A	○ SA
Navigating through the menus and toolbars is easy to do.	○ SD	○ D	○ N	○ A	○ SA
This software is engaging.	○ SD	○ D	○ N	○ A	○ SA
The contents of the menus and the toolbars match my needs.	○ SD	○ D	○ N	○ A	○ SA
Getting started with this version of the	○ SD	○ D	○ N	○ A	○ SA



software is easy.					
This software is flexible.	○ SD	○ D	○ N	○ A	○ SA
Finding the options that I want in the menus and toolbars is easy.	○ SD	○ D	○ N	○ A	○ SA
It is easy to make the software do exactly what I want.	○ SD	○ D	○ N	○ A	○ SA
Discovering new features is easy.	○ SD	○ D	○ N	○ A	○ SA
I get my word processing tasks done quickly with this software.	○ SD	○ D	○ N	○ A	○ SA
This software is satisfying to use.	○ SD	○ D	○ N	○ A	○ SA

9. If you have any other comments with respect to Microsoft Word 2000 or word processing in general, feel free to make them here:

Department of Computer Science and
Knowledge Media Design Institute
University of Toronto

Last modified:

MSWord Study: Questionnaire #2

First name: □

Lastname: □

Date: □

This questionnaire is to be completed the first day[62] that you start to use MSWord Personal.

The questions in this questionnaire will use the following response scale:

SD = Strongly Disagree
D = Disagree
N = Neutral
A = Agree
SA = Strongly Agree

1. With respect to the version of Microsoft Word currently installed on your machine, Microsoft Word Personal, please indicate the extent to which you agree or disagree with the following statements:

This software is easy to use.	○ SD	○ D	○ N	○ A	○ SA
I am in control of the contents of the menus and toolbars.	○ SD	○ D	○ N	○ A	○ SA
I will be able to learn how to use all that is offered in this software.	○ SD	○ D	○ N	○ A	○ SA
Navigating through the menus and toolbars is easy to do.	○ SD	○ D	○ N	○ A	○ SA
This software is engaging.	○ SD	○ D	○ N	○ A	○ SA
The contents of the menus and the toolbars	○ SD	○ D	○ N	○ A	○ SA

[62] Note that Questionnaires #3, #4, #5, and #6 (Q3, Q4, Q5, and Q6) contain identical questions to Questionnaire #2 (Q2), which is why they are not shown in this appendix. They differ only in terms of the statement that indicates when, with respect to starting to use MSWord Personal, the questionnaires are to be completed.

match my needs.					
Getting started with this version of the software is easy.	○ SD	○ D	○ N	○ A	○ SA
This software is flexible.	○ SD	○ D	○ N	○ A	○ SA
Finding the options that I want in the menus and toolbars is easy.	○ SD	○ D	○ N	○ A	○ SA
It is easy to make the software do exactly what I want.	○ SD	○ D	○ N	○ A	○ SA
Discovering new features is easy.	○ SD	○ D	○ N	○ A	○ SA
I get my word processing tasks done quickly with this software.	○ SD	○ D	○ N	○ A	○ SA
This software is satisfying to use.	○ SD	○ D	○ N	○ A	○ SA

2. In Microsoft Word Personal you have the ability to add/delete functions to/from your personal interface. With respect to the mechanism used for adding and deleting functions please indicate the extent to which you agree or disagree with the following statements:

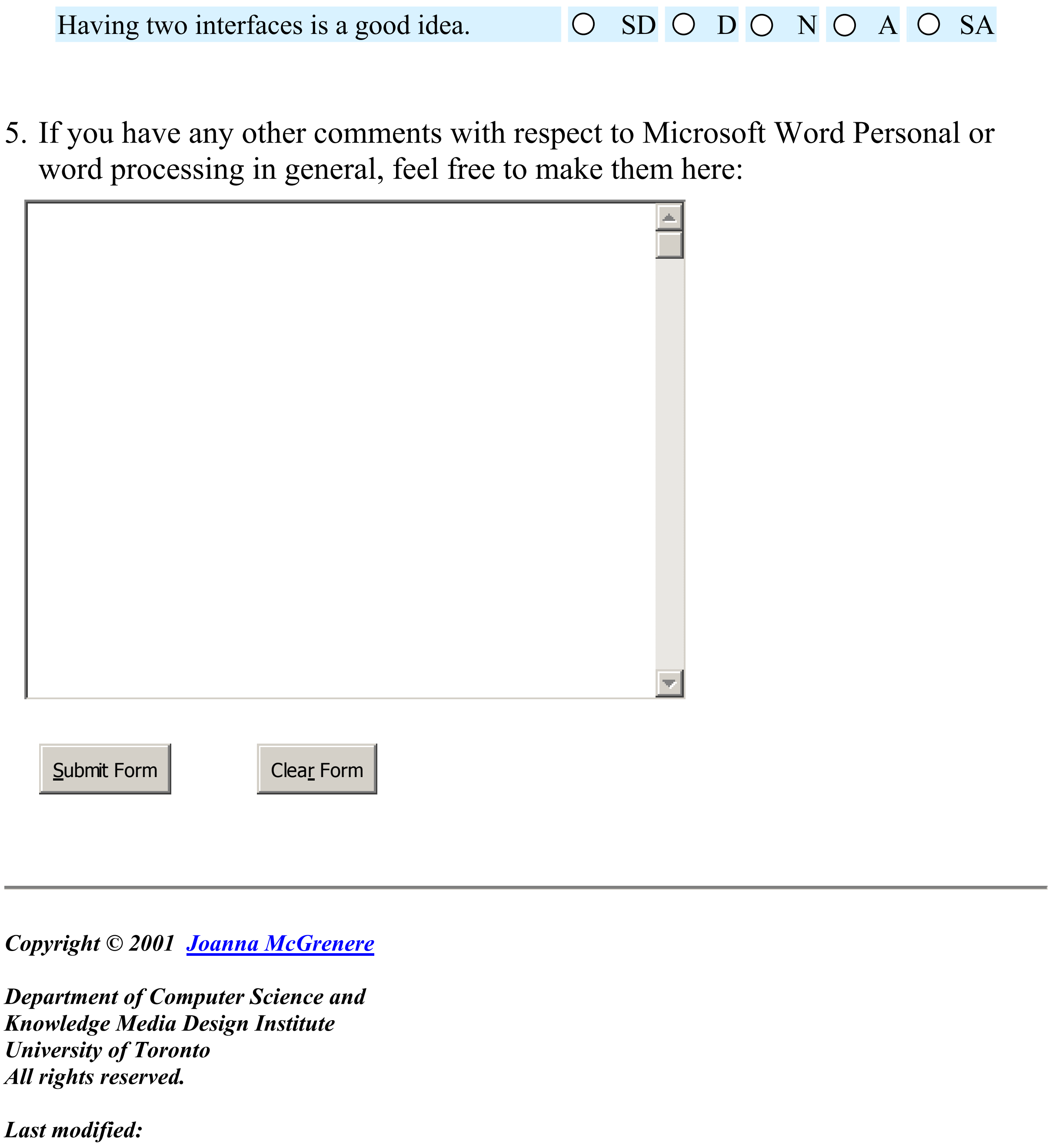
This mechanism is easy to use.	○ SD	○ D	○ N	○ A	○ SA
This mechanism is intuitive.	○ SD	○ D	○ N	○ A	○ SA
This mechanism is flexible - I can modify my personal interface so it is exactly how I want it.	○ SD	○ D	○ N	○ A	○ SA

3. In Microsoft Word Personal you have both a personal interface and the Full Interface. With respect to the mechanism used to switch between interfaces please respond to the following statement:

This mechanism is easy to use.	○ SD	○ D	○ N	○ A	○ SA

4. With respect to the concept of having two interfaces, please respond to the following statements:

This concept is easy to understand.	○ SD	○ D	○ N	○ A	○ SA



Having two interfaces is a good idea. ○ SD ○ D ○ N ○ A ○ SA

5. If you have any other comments with respect to Microsoft Word Personal or word processing in general, feel free to make them here:

Department of Computer Science and
Knowledge Media Design Institute
University of Toronto

Last modified:

MSWord Study: Questionnaire #7

First name: □

Lastname: □

Date: □

This questionnaire is to be completed one week after the Second Meeting.

1. For the past week or so you have been using Microsoft Word 2000 once again. With respect to Microsoft Word 2000, please indicate the extent to which you agree or disagree with the following statements:

SD = Strongly Disagree
D = Disagree
N = Neutral
A = Agree
SA = Strongly Agree

This software is easy to use.	○ SD	○ D	○ N	○ A	○ SA
I am in control of the contents of the menus and toolbars.	○ SD	○ D	○ N	○ A	○ SA
I will be able to learn how to use all that is offered in this software.	○ SD	○ D	○ N	○ A	○ SA
Navigating through the menus and toolbars is easy to do.	○ SD	○ D	○ N	○ A	○ SA
This software is engaging.	○ SD	○ D	○ N	○ A	○ SA
The contents of the menus and the toolbars match my needs.	○ SD	○ D	○ N	○ A	○ SA
Getting started with this version of the software is easy.	○ SD	○ D	○ N	○ A	○ SA
This software is flexible.	○ SD	○ D	○ N	○ A	○ SA
Finding the options that I want in the menus and toolbars is easy.	○ SD	○ D	○ N	○ A	○ SA
It is easy to make the software do exactly what I want.	○ SD	○ D	○ N	○ A	○ SA

Discovering new features is easy.	○ SD	○ D	○ N	○ A	○ SA
I get my word processing tasks done quickly with this software.	○ SD	○ D	○ N	○ A	○ SA
This software is satisfying to use.	○ SD	○ D	○ N	○ A	○ SA

2. If you have any other comments with respect to Microsoft Word 2000 or word processing in general, feel free to make them here:

Submit Form   Clear Form

Last modified:

MSWord Study: Questionnaire #8

First name: □
Lastname: □
Date: □

This questionnaire is to be completed 2 weeks after the Second Meeting.

You have been exposed to two different versions of Microsoft Word:

Microsoft Word 2000:	You were using this version before you particpated in this study and you have been using it again for the past week or so.
Microsoft Word Personal:	You were using this version for the first four weeks of this study.

1. If you could choose only one of the versions to continue using, which would it be?

   ○ Microsoft Word 2000 ○ Microsoft Word Personal

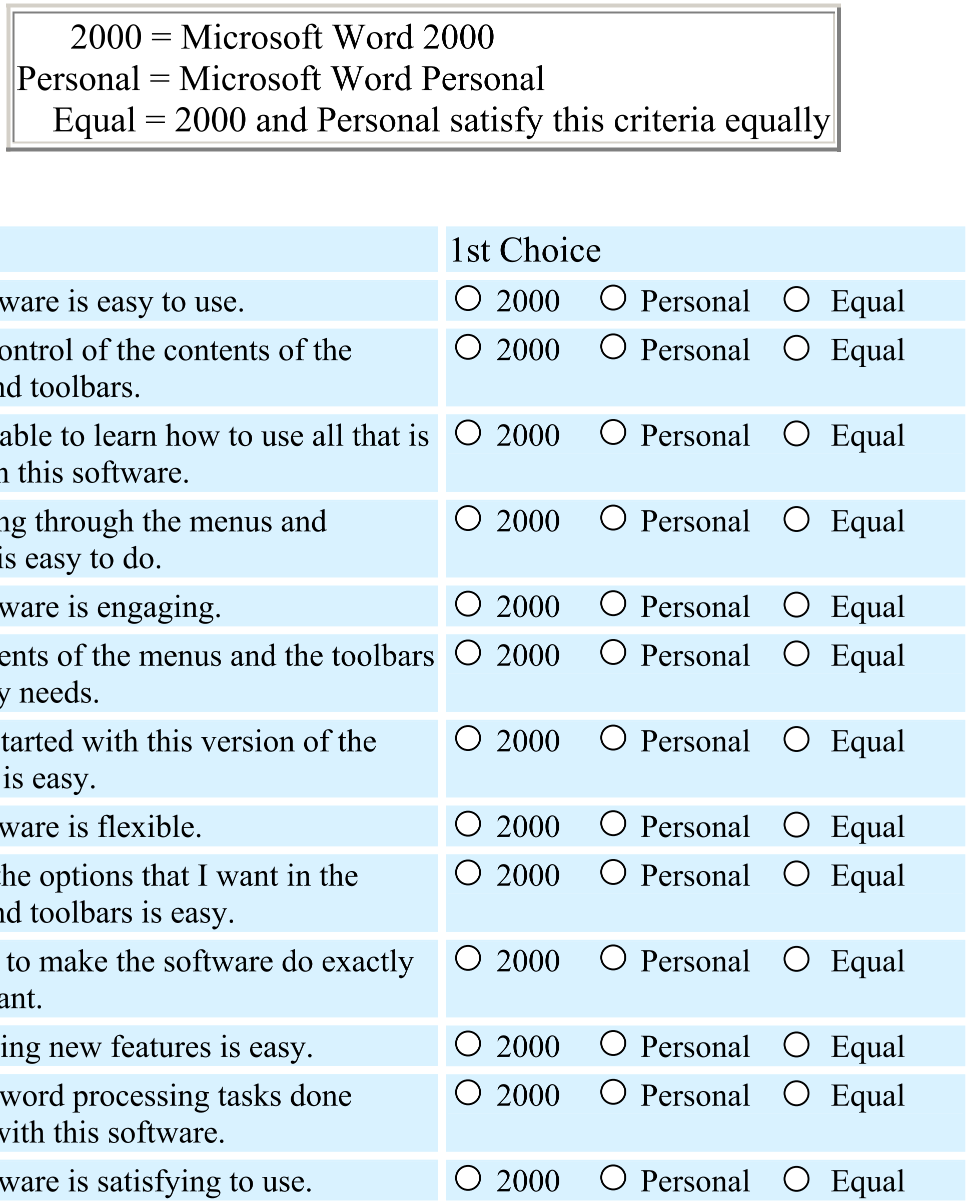
2. What particular aspect(s) of Microsoft Word 2000 did you *like*?

3. What particular aspect(s) of Microsoft Word 2000 did you *dislike*?

4. What particular aspect(s) of Microsoft Word Personal did you *like*?

5. What particular aspect(s) of Microsoft Word Personal did you *dislike*?

6. There are a number of criteria listed below. Please select the version that would be your 1st choice *according to each of the criteria*. If you really cannot make a choice for a given criteria please select "Equal".



2000 = Microsoft Word 2000
Personal = Microsoft Word Personal
Equal = 2000 and Personal satisfy this criteria equally

Criteria	1st Choice
This software is easy to use.	○ 2000 ○ Personal ○ Equal
I am in control of the contents of the menus and toolbars.	○ 2000 ○ Personal ○ Equal
I will be able to learn how to use all that is offered in this software.	○ 2000 ○ Personal ○ Equal
Navigating through the menus and toolbars is easy to do.	○ 2000 ○ Personal ○ Equal
This software is engaging.	○ 2000 ○ Personal ○ Equal
The contents of the menus and the toolbars match my needs.	○ 2000 ○ Personal ○ Equal
Getting started with this version of the software is easy.	○ 2000 ○ Personal ○ Equal
This software is flexible.	○ 2000 ○ Personal ○ Equal
Finding the options that I want in the menus and toolbars is easy.	○ 2000 ○ Personal ○ Equal
It is easy to make the software do exactly what I want.	○ 2000 ○ Personal ○ Equal
Discovering new features is easy.	○ 2000 ○ Personal ○ Equal
I get my word processing tasks done quickly with this software.	○ 2000 ○ Personal ○ Equal
This software is satisfying to use.	○ 2000 ○ Personal ○ Equal

7. If you have any other comments with respect to either version of Microsoft Word or your involvement in this study, feel free to make them here:

Last modified:

Appendix O: Study Two Interview

Final Interview Questions

Questions specific to individual participant. Questions were generated based on reviewing the participants logging and questionnaire data prior to the interview.

As you know your interactions with MSWord were being logged during this study. The first few questions I am going to ask are about some things I observed from the log files, in particular, the log files from the first 4 weeks of the study, while you were using MSWord Personal.

1. I can see from your log files that you spent roughly _____% of your time in your PI and _____% in the FI. You actually switched from your PI to the FI _______ times. Does this sound about right?

    a. How would you describe your approach to using the two interfaces and the personalizing mechanism?

    If participant doesn't elaborate in part (a) here are some probing questions.

    [For those who switched at least once]

    b. Why did you to switch to the FI? What circumstances prompted you to switch?

c. I also noticed that you switched from the FI to the PI _______ times. Why did you switch back?

d. Why didn't you switch back to your PI every time you've switched to the FI?

[For those who never switched]

e. Why did you not switch to the FI?

2. I see you added _____ functions to your PI while you were using MSWord Personal. Can you describe the process that you went through in terms of selecting whether or not to add a given function? For example, why would you choose to add one function and not another?

3. You added _____ functions to your PI yet I noticed that you *did not use* any of these during the 4 weeks that you were using MSWord Personal. Can you tell me why you added them to your PI? [Name the actual functions for the participant.]

4. There were also _____ functions that you *did use* but that you chose *not* to add to your personal interface. Can you tell me why? [Name the actual functions for the participant.]

5. Deleting functions:

   [For those who did not delete any functions]

   a. I noticed that although you added a bunch of functions to your PI you never deleted any functions. Can I ask you why? Do you think you would ever delete a function from your PI?

   [For those who did delete functions]

   b. I noticed that you deleted _____ functions from your PI. [Name the deleted functions] Can you tell me why you deleted these?

Now, I'm leaving questions about the logging. Here's a hypothetical question for you.

6. Let's just say that I were to leave MSWord Personal on your machine today but we could reconfigure it somewhat to better suit your needs – which of the following four options would your choose:

   1. Have only your PI – so no FI. (The PI would still be able to include any functions that you wanted but you wouldn't be able to switch to the FI and use functions from there.)

   2. Have only the FI – so no PI.

3. Have both your PI and the FI exactly as you used it during the first 4 weeks of the study.
4. Have both your PI and the FI but instead of MSWord opening with the PI being displayed, it would open with the FI displayed (but you could then switch your PI if you chose to).

Explain why?

7. From your final Comparison Questionnaire I can see that you had a general preference for ______________. Can you tell me the main reason why chose this one?

8. 3-way choice:

   a. With respect to Word 2000 – the version of Word you have been using for the last two weeks – did you notice the changing menus – they are initially short and you can make them long by clicking on the double arrow?

      _____ Yes   _____ No

   b. Do you know how these menus work?   [they are adapting]

      _____ Yes   _____ No

c. Did you know that you could turn these menus off so that you just have fixed long menus that contain all the functions? [Explain if they are confused.]

_____ Yes   _____ No

d. Knowing this, let me ask you – if you had the following three word processor options

- MSWord Personal
- MSWord 2000 *with* the changing menus, and
- MSWord 2000 *without* the changing menus

Which would you choose to continue working with on your machine? What would be your first, second, and third choice?

_____ Personal   ____ 2000 adaptive on   ____ 2000 adaptive off

e. Why have you ranked them this way?

9. Can you comment on the issue of learning – in particular how do you think the three previously mentioned interfaces would impact your ability to learn new functions in Word? Do you think one of the interfaces would allow you to learn more functions than the others? Or would there be no difference?

10. On the preliminary questionnaire (the questionnaire you filled out before you even began participating in this study) there was a statement that read:

> I would prefer to have only the functions I use in the menus/toolbars - the other functions could be "tucked" away in the event that I might need them someday.

You _________ [SD, D, N, A, SA] to this statement at that time

a. Do you think MSWord Personal provides an effective way of tucking away unused functions? If not, can you suggest another way to do this?

b. Now that you have had an opportunity to work with MSWord Personal – which does allow unused functions to be tucked away – would you change your preference?

The last few questions have to do with the whole issue of participating in a study.

11. During the last 6 weeks or so while you have been involved in this study, to what extent were you aware on a daily basis that you were participating in a study?

So which of the following best describes your awareness:

1 = minimally aware: apart from when I was reminded to do the questionnaires, participating in the study barely crossed my mind

2 = somewhat aware: occasionally it would occur to me that someone would be looking at my data and analyzing my activities

3 = very aware: a significant amount of the time I used MSWord I was conscious that someone would be looking at my data and analyzing my activities

12. Why did you participate in this study? What motivated you to participate?

13. Just out of curiousity, to what extent did the $100 gift certificate influence you to participate in this study? The reason why it is helpful for me to know this is that I will likely be running other studies in the future and soliciting participants can sometimes be challenging (so please be as honest as you can!)

So which of the following best describes the influence on you:

1 = no influence: I would have participated without the $100

2 = somewhat of an influence: I may have participated without the $100

3 = strong influence: I would not have participated without the $100

14. What about the $100 prize for completing the questionnaires on time? Did this have an influence on your diligence to completing the questionnaires?

1 = no influence

2 = somewhat of an influence

3 = strong influence

15. How did you find the experience of participating in this study?

16. Is there anything I could have done to make the experience easier or more enjoyable for you?

17. Would you participate in another study like this again? By “this” I mean a study where someone installs something new on your machine for a short period of time and solicits feedback along the way.

___ No ___ Probably Not ___ Maybe ___Probably ___Yes

18. Do you have any other comments – anything at all about the versions of the word processor or your participation in this study?

Appendix P: MSTracker Grammar and Output Files

MSTracker Grammar

Although we did not use a parser generator, we did document the grammar of the MSTracker log file using a BNF-like syntax. The grammar was essentially hard coded into the parser.

The grammar for selecting a toolbar item or an interface is shown below.

toolbar <function>:

MOUSE
MOUSE IDLE
MENUSTART

```
    {<TB mousedown rec> | <TB mouseidle rec>} <MENUSTART rec> [A] <any record> <MENUEND rec>
    |
    <MENUSTART rec> <any record> (<TB mouseidle rec>|<interface dropdown rec>) [A] <any record>
    <MENUEND rec>
    |
    {<interface dropdown rec>} <MENUSTART rec> <any record> <interface selection rec> <any record>
    <MENUEND rec>
    |
    <MENUSTART rec> <any record> <interface dropdown rec> <any record> <interface selection rec> <any
    record> <MENUEND rec>


IF <TB roles> is one of  {“combo box” | “drop down”} at [A] there may be:
        If “drop down”:
        <MENUPOPUSTART rec><TB mousedown rec>
        If  “combo box”:
        <TB mousedown rec>

<interface dropdown rec>:
        “MOUSE” “Interfaces” “drop down” “WM_LBUTTONDOWN CLIENT”
        |
        “MOUSE IDLE” “Interfaces” “drop down” “ “
<interface selection rec>:
        “MOUSE” <interface> “list item” “WM_LBUTTONUP CLIENT”

<TB mousedown rec>:
        “MOUSE” <TB functions> <TB roles> “WM_LBUTTONUP CLIENT”
<TB functions>:
        “Style:” | “Font:” | “Font Size:” | “Bold” | “Italic” | “Underline” | etc.
<TB roles>:
        “combo box” | “push button” | “drop down” | “grid drop down” | “menu button”

<TB mousidle rec>:
        “MOUSE IDLE” <TB functions> <TB roles> blank

<MENUSTART rec>:
        “MENUSTART” “Menu Bar” “menubar” “<unknown>”

<MENUEND rec>:
```

“MENUEND” “Menu Bar” “menubar” “<unknown>”

Processed Output Files from MSTracker

Here are snippets of output files. There are two files: a processed log file and a summary stats file. Annotation is in bold.

File 1: Processed log file

This is an output file from the parser that has been put though one additional level of processing to add interface information to the log file:

The first 3 columns come directly from the original MS Access file: log item, date, timestamp. The 4th column comes from the additional processing level. The format is “w,xyz” where “w” is the active interface (1 is minimal, 2 is personal, 3 is default), and “xyz” show the availability of this logged function within interfaces 1,2,3 respectively. A “0” indicates that the function was not available in the interface whereas a “1” indicates available.

The rest of the output line comes from the parser.

```
5 8/29/2000 10:09:50 1,111 Switch TO Word [document1 10microsoft word]
12 8/29/2000 10:09:50 1,111 Launch Word [reminderstart] ******
24 8/29/2000 10:09:53 1,111 [keyboard]
27 8/29/2000 10:09:55 1,111 [verticalScroll]
34 8/29/2000 10:09:55 1,111 [verticalScroll]
```

switches to default interface

```
134 8/29/2000 10:10:19 3,111 [interfacesTB] [default interface]
```

undo is only available in the default interface as shown by the 3,001 pattern

```
156 8/29/2000 10:10:21 3,001 [editMenu] [undo typing]
217 8/29/2000 10:10:29 3,111 [keyboard]
218 8/29/2000 10:10:32 3,111 [align leftTB]
229 8/29/2000 10:10:34 3,111 [align leftTB]
244 8/29/2000 10:10:42 3,111 [keyboard]
245 8/29/2000 10:10:46 3,111 [keyboard]
250 8/29/2000 10:10:48 3,111 [keyboard]
255 8/29/2000 10:10:54 3,111 [align leftTB]
266 8/29/2000 10:10:55 3,001 [decrease indentTB]
277 8/29/2000 10:10:56 3,001 [decrease indentTB]
295 8/29/2000 10:10:58 3,111 [verticalScroll]
328 8/29/2000 10:11:49 3,111 Switch FROM Word
353 8/29/2000 10:11:56 3,111 Switch TO Word [document1 microsoft word]
378 8/29/2000 10:12:02 3,001 [editMenu] [undo paste]
396 8/29/2000 10:12:04 3,111 [verticalScroll]
404 8/29/2000 10:12:17 3,111 Switch FROM Word
442 8/29/2000 10:12:35 3,111 Switch TO Word [document1 microsoft word]
445 8/29/2000 10:12:35 3,111 Switch TO Word [document1 microsoft word]
447 8/29/2000 10:12:35 3,111 Switch TO Word [document1 microsoft word]
474 8/29/2000 10:16:28 3,111 Switch FROM Word
576 8/29/2000 10:17:05 3,111 Switch TO Word [document1 microsoft word]
579 8/29/2000 10:17:05 3,111 Switch TO Word [document1 microsoft word]
```

```
581 8/29/2000 10:17:05 3,111 Switch TO Word [document1 microsoft word]
651 8/29/2000 10:19:36 3,111 [verticalScroll]
659 8/29/2000 10:19:42 3,111 [verticalScroll]
670 8/29/2000 10:19:56 3,111 [saveTB]
686 8/29/2000 10:20:01 3,111 [keyboard]
784 8/29/2000 10:20:09 3,111 [verticalScroll]
810 8/29/2000 10:20:24 3,111 Switch FROM Word
812 8/29/2000 10:20:27 3,111 Switch TO Word [milagro.doc microsoft word]
816 8/29/2000 10:20:27 3,111 Switch TO Word [milagro.doc microsoft word]
848 8/29/2000 10:21:02 3,111 [boldTB]
858 8/29/2000 10:21:05 3,111 [centerTB]
869 8/29/2000 10:21:10 3,001 [bordersTB]
880 8/29/2000 10:21:11 3,001 [bordersTB] [outside borders]
916 8/29/2000 10:21:16 3,001 [bordersTB] [outside borders]
954 8/29/2000 10:21:22 3,001 [bordersTB]
969 8/29/2000 10:21:26 3,001 [bordersTB] [no border]
1006 8/29/2000 10:21:33 3,001 [bordersTB] [no border]
1042 8/29/2000 10:21:37 3,111 [centerTB]
1059 8/29/2000 10:21:41 3,111 [font size:TB] [16]
1097 8/29/2000 10:23:01 3,111 [verticalScroll]
1110 8/29/2000 10:23:08 3,111 [fileMenu] [page setup...]
1187 8/29/2000 10:23:25 3,111 [verticalScroll]
1195 8/29/2000 10:23:27 3,111 [saveTB]
1205 8/29/2000 10:23:42 3,111 [verticalScroll]
1212 8/29/2000 10:23:44 3,111 Switch FROM Word
1376 8/29/2000 10:24:42 3,111 Switch TO Word [milagro.doc microsoft word]
1378 8/29/2000 10:24:48 3,111 [saveTB]
1391 8/29/2000 10:24:50 3,111 [verticalScroll]
1413 8/29/2000 10:25:40 3,111 [saveTB]
1425 8/29/2000 10:26:08 3,111 [printTB]
1439 8/29/2000 10:26:11 3,101 [fileMenu] [print...]
1487 8/29/2000 10:27:51 3,111 [saveTB]
1495 8/29/2000 10:27:58 3,111 [printTB]
1507 8/29/2000 10:28:11       [CloseButton]
1521 8/29/2000 10:28:15 3,111 Switch FROM Word
1549 8/29/2000 10:32:05 3,111 Switch TO Word [microsoft word]
```

word launches in minimal interface

```
1557 8/29/2000 10:32:05 1,111 Launch Word [reminderstart] ******
1623 8/29/2000 10:32:22 1,111 [align leftTB]
1645 8/29/2000 10:33:16 1,111 [saveTB]
1671 8/29/2000 10:33:26       [CloseButton]
1704 8/29/2000 10:35:03 1,111 [verticalScroll]
1713 8/29/2000 10:35:09 1,111 [font size:TB] [14]
1740 8/29/2000 10:35:12 1,111 [saveTB]
1749 8/29/2000 10:35:15 1,111 [verticalScroll]
1757 8/29/2000 10:35:17 1,111 [printTB]
1764 8/29/2000 10:35:22       [CloseButton]
1777 8/29/2000 10:35:24 1,111 Switch FROM Word
1804 8/29/2000 10:37:03 1,111 Launch Word [microsoft word] ******
1810 8/29/2000 10:37:05 1,111 [printTB]
1826 8/29/2000 10:37:07 1,101 [fileMenu] [print...]
1850 8/29/2000 10:37:10       [CloseButton]
1857 8/29/2000 10:37:16       [CloseButton]
1870 8/29/2000 10:37:28 1,111 Switch FROM Word
```

File 2: Summary stats file

This file gets produced for each user.

```
1 Minimal Interface.int
2 Joanna's Interface.int


************joanna\Jul11.out

Time in interface 1: 00:01:25
Time in interface 2: 00:42:46
Time in interface 3: 00:00:00

61     111 [keyboard]
14     111 Switch TO Word
7      111 Switch FROM Word
4      011 [decrease indentTB]
4      011 [increase indentTB]
3      111 Launch Word
1      111 [fileMenu] [1
1      101 [fileMenu] [open...]
1      111 [interfacesTB] [2
1      011 [viewMenu] [ruler]
```

First column gives number of occurrences, second column is the same xyz from the output file (File 1 shown above).

```
************joanna\Jul12.out

Time in interface 1: 00:00:16
Time in interface 2: 00:00:00
Time in interface 3: 00:00:00

2      111 Switch FROM Word
2      111 Switch TO Word


************joanna\Aug30.out

Time in interface 1: 00:01:59
Time in interface 2: 01:55:58
Time in interface 3: 00:00:00

120    111 [keyboard]
23     111 Switch TO Word
11     111 Switch FROM Word
11     011 [numberingTB]
10     111 [verticalScroll]
5      011 [style:TB]
2      111 [bulletsTB]
2      011 [zoom:TB]
1      111 [PrintLayoutViewButton]
1      011 [decrease indentTB]
```

```
1     111 [fileMenu] [save as...]
1     011 [format painterTB]
1     111 [formatMenu] [font...]
1     011 [formatMenu] [paragraph...]
1     011 [increase indentTB]
1     111 [interfacesTB] [2
1     011 [toolsMenu] [word count...]


************joanna\Aug31.out

Time in interface 1: 00:00:10
Time in interface 2: 06:05:58
Time in interface 3: 00:00:00

102   111 [keyboard]
51    111 Switch TO Word
21    111 Switch FROM Word
12    111 [verticalScroll]
2     111 [interfacesTB] [2
2     111 [minimizeRestore]
2     111 [minimize]
2     011 [zoom:TB]
1     111 [NormalViewButton]
1     111 [fileMenu] [1
1     111 [fileMenu] [print...]
1     011 [formatMenu] [bullets and numbering...]
1     011 [formatMenu] [columns...]
1     011 [formatMenu] [paragraph...]
1     111 [horizontalScroll]
1     011 [numberingTB]
1     111 [printTB]
1     011 [redo
1     011 [style:TB]
1     011 [toolsMenu] [word count...]
```

A number of the daily summaries have been omitted for brevity.

```
************TOTAL (function availability for most recent interface files)

Number of logs process: 21

Time in interface 1: 06:25:16
Time in interface 2: 13:56:00
Time in interface 3: 00:37:16
```

Format: 1st col is total number of occurrences, 2nd col is the xyz, and then the function. On the second line you can see a big bit pattern. There is one bit per log and a 1 indicates that the function was used in that log. The number at the end just gives the total number of logs in which the function was used (add the 1's up).

```
1269 111 [keyboard]
            100111111111111110111  18
472 111 Switch TO Word
```

```
            111111111111111111111  21
318 111 Switch FROM Word
            111111111111111111111  21
51    111 [verticalScroll]
            000010100010001100101  7
16    111 Launch Word
            100011101110100001111  12
16    111 [fileMenu] [print...]
            000111000000110110011  9
16    011 [style:TB]
            000000000000001110000  3
15    011 [format painterTB]
            000000000000001010000  2
15    111 [minimize]
            000100101000000100101  6
13    111 [interfacesTB] [2
            101001101110011110001  12
12    011 [decrease indentTB]
            100000000100011010000  5
12    111 [minimizeRestore]
            000100101000000100001  5
12    011 [numberingTB]
            000000000000001100000  2
11    111 [printTB]
            000100100100010100001  6
8     011 [increase indentTB]
            100000100000011000000  4
6     111 [font size:TB]
            000001000010000010000  3
5     011 [editMenu] [paste special...]
            000000101000000000000  2
5     101 [fileMenu] [open...]
            100000000101000010100  5
5     111 [fileMenu] [save as...]
            000010000010001010000  4
5     011 [formatMenu] [paragraph...]
            000000000000001110000  3
5     011 [zoom:TB]
            000000000010001100000  3
4     111 [bulletsTB]
            000000000000001010000  2
4     111 [fileMenu] [1
            100000000000000101001  4
3     111 [NormalViewButton]
            000010100000000100000  3
3     011 [bordersTB]
            000000100000000010000  2
3     111 [formatMenu] [font...]
            000000000000011010000  3
3     011 [new blank documentTB]
            000000100000000010000  2
3     001 [toolsMenu] [track changes menuMenu] [compare documents...]
            000000000001000000000  1
3     011 [toolsMenu] [word count...]
            000000000000001100001  3
3     111 [viewMenu] [header and footer]
            000000000000000010000  1
```

```
2     111 [PrintLayoutViewButton]
            000000000000001010000  2
2     111 [fileMenu] [3
            000010000000000001000  2
2     101 [fileMenu] [new...]
            000000010000000001000  2
2     011 [formatMenu] [bullets and numbering...]
            000000000000000110000  2
2     011 [print previewTB]
            000000000010000010000  2
2     011 [redo
            000000000000000110000  2
2     011 [toolsMenu] [track changes menuMenu] [accept or reject
changes...]
            000000001010000000000  2
2     011 [viewMenu] [ruler]
            100000000000000010000  2
1     111 [align leftTB]
            000000000010000000000  1
1     111 [align rightTB]
            000000000100000000000  1
1     111 [fileMenu] [2
            000000000000000000001  1
1     111 [font:TB]
            000010000000000000000  1
1     011 [formatMenu] [columns...]
            000000000000000100000  1
1     111 [horizontalScroll]
            000000000000000100000  1
1     011 [insertMenu] [break...]
            000000000000000010000  1
1     111 [interfacesTB] [default interface]
            000000000001000000000  1
1     011 [open...TB]
            000000000001000000000  1

TOTAL number of different functions used: 43
```

Everything is counted as a " function" except Switch to Word, Switch from Word, Launch Word, and keyboard.

This next section lists all the functions available in the user's PI and gives an occurrence count. It's supposed to answer the question Do users use what they ask for?

```
Functions available in personal interface:

3      [new blank documentTB]
1      [open...TB]
11     [printTB]
2      [print previewTB]
0      [spelling and grammar...TB]
15     [format painterTB]
0      [undo
2      [redo
```

```
0      [insert table...TB]
0      [columnsTB]
0      [drawingTB]
0      [document mapTB]
0      [show allTB]
5      [zoom:TB]
0      [microsoft word helpTB]
16     [style:TB]
1      [font:TB]
6      [font size:TB]
1      [align leftTB]
0      [centerTB]
1      [align rightTB]
0      [justifyTB]
12     [numberingTB]
4      [bulletsTB]
12     [decrease indentTB]
8      [increase indentTB]
3      [bordersTB]
5      [fileMenu] [save as...]
0      [fileMenu] [page setup...]
0      [fileMenu] [print preview]
16     [fileMenu] [print...]
5      [editMenu] [paste special...]
2      [viewMenu] [ruler]
3      [viewMenu] [header and footer]
0      [viewMenu] [footnotes]
1      [insertMenu] [break...]
0      [insertMenu] [page numbers]
0      [insertMenu] [symbol...]
0      [insertMenu] [footnote...]
0      [insertMenu] [caption...]
0      [insertMenu] [cross-reference...]
0      [insertMenu] [index and tables...]
3      [formatMenu] [font...]
5      [formatMenu] [paragraph...]
2      [formatMenu] [bullets and numbering...]
0      [formatMenu] [borders and shading...]
1      [formatMenu] [columns...]
0      [formatMenu] [style...]
0      [formatMenu] [object...]
3      [toolsMenu] [word count...]
0      [toolsMenu] [autocorrect...]
0      [toolsMenu] [track changes menuMenu] [highlight changes...]
2      [toolsMenu] [track changes menuMenu] [accept or reject changes...]
0      [toolsMenu] [macroMenu] [macros...]
0      [toolsMenu] [macroMenu] [record new macro...]
0      [toolsMenu] [macroMenu] [security...]
0      [toolsMenu] [macroMenu] [visual basic editor]
0      [toolsMenu] [macroMenu] [microsoft script editor]
0      [toolsMenu] [templates and add-ins...]
0      [toolsMenu] [customize...]
0      [toolsMenu] [options...]
0      [tableMenu] [toolbar 32779Menu] [table...]
0      [tableMenu] [merge cells]
0      [tableMenu] [split cells...]
0      [tableMenu] [split table]
```

```
0     [tableMenu] [table autoformat]
0     [tableMenu] [autofitMenu] [distribute rows evenly]
0     [tableMenu] [autofitMenu] [distribute columns evenly]
0     [tableMenu] [toolbar 32781Menu] [text to table...]
0     [tableMenu] [toolbar 32781Menu] [table to text...]
0     [tableMenu] [sort...]
0     [tableMenu] [hide gridlines]
0     [helpMenu] [microsoft word help]
0     [helpMenu] [what's this?]
0     [helpMenu] [about microsoft word]
51    [verticalScroll]
0     [SelectBrowseObjectButton]
3     [NormalViewButton]
0     [WebLayoutViewButton]
2     [PrintLayoutViewButton]
0     [OutlineViewButton]
```

Everything from this point down is not an option – these are available in all interfaces.

```
1     [horizontalScroll]
0     [interfacesTB] [1 minimal interface]
13    [interfacesTB] [2
1     [interfacesTB] [default interface]
15    [minimize]
12    [minimizeRestore]
0     [maximize]
1269  [keyboard]
318   Switch FROM Word
472   Switch TO Word
16    Launch Word
4     [fileMenu] [1
1     [fileMenu] [2
2     [fileMenu] [3
0     [fileMenu] [4
0     [windowMenu] [1
0     [windowMenu] [2
0     [windowMenu] [3
0     [windowMenu] [4
```

Appendix Q: Multiple Interfaces Implementation

The information provided in this appendix is intended to provide some of the implementation details for the multiple interfaces prototype. The prototype had two main requirements. The first was to simulate the multiple interface features for toggling between the *Full Interface* and *Personal Interface* and for modifying the personal interface by adding/deleting functions to/from it. The second requirement was to maintain state information across MSWord sessions. These requirements were satisfied by writing code in Visual Basic for Applications that executed within MSWord at various times, and which maintained state information in external, flat files that employed a format chosen specifically for this task.

MSWord is programmable through Visual Basic for Applications (VBA). VBA exposes application programming interfaces that can be implemented or invoked with code written in the Visual Basic programming language. This form of programming is commonly known as macro programming. In the next sections we describe the MSWord template facility which is where VBA code can reside, the Visual Basic Editor which is where VBA code is created/edited, and lastly some of the VBA code that was used in our implementation. We show the VBA code for the mechanism that saves state information to a flat file, for the mechanism that uses the information to initialize the prototype, for the mechanism for toggling between interfaces, and for the mechanism for personalizing the interface.

Although we only provide some of the code, the detail provided here should be sufficient to help a programmer familiar with VBA begin to replicate our implementation.

MSWord Templates

VBA macro code can be located in one of three locations:

1) a specific document[63]

2) a document template

3) a global template

We chose the third alternative for a number of reasons. Our multiple interfaces prototype essentially provided the appearance of more than one interface between which the user could toggle. The availability of these multiple interfaces was supposed to be independent of documents – i.e., the multiple interfaces needed to be available regardless of what document was open and even if there was no document open at all. For this reason it did not make sense to attach our code to a particular document (option 1 above). It also meant that the code could

[63] Documents that include macros are one common source of viruses.

not be attached to a document template (option 2) because different documents often use different templates and a document template is only loaded/active when a document is created based on that template. Clearly this did not meet our needs.

The indented paragraphs below come directly from the MSWord help system. (Once in help, select the **Contents** tab, **Templates and Wizards**, **About Templates**.)

> **Overview of templates**
>
> Every Microsoft Word document is based on a template. A template determines the basic structure for a document and contains document settings such as AutoText entries, fonts, key assignments, macros, menus, page layout, special formatting, and styles. The two basic types of templates are global templates and document templates. Global templates, including the Normal template, contain settings that are available to all documents. Document templates, such as the memo or fax templates in the **New** dialog box, contain settings that are available only to documents based on that template. For example, if you create a memo using the memo template, the memo can use the settings from both the memo template as well as the settings in any global template. Word provides a variety of document templates, and you can create your own document templates.
>
> **Normal Template**
>
> The Normal template is a general-purpose template that you can use for any document. When you start MSWord or click on the **New Blank Document**, Word creates a new blank document that is based on the Normal template. You can modify this template to change the default document formatting (e.g., page margins) and content.
>
> You should store the Normal template in the Templates folder or in the User Templates or Workgroup Templates file location you specified on the **File Locations** tab (**Tools** menu, **Options** command). If Word can’t find the Normal template in any of these locations or in your Word program folder, it creates a new Normal template with standard Word document settings.
>
> **Working with global templates**
>
> When you work on a document, you can typically use only the settings stored in the template attached to the document or in the Normal template. To use any such items that are stored in another template, you can load the other template as a global template. After you load a template, items stored in that template are available to any document during the remainder of the Word session. When you're finished with the items, be sure to unload the template to conserve system resources.
>
> Add-ins and templates that you load are unloaded when you close Word. To load an add-in or template each time you start Word, copy the add-in or template to the Startup folder, whose location is specified on the **File Locations** tab (**Tools** menu, **Options** command).

As might be obvious from this excerpt, the information about templates available in the online help is somewhat cryptic. Much of what we learned about templates was learned through trial and error.

Given that the multiple interfaces code needed to be stored in a global template we had one of two options, namely to store the code in the Normal template or to create our own global template. We decided to create our own global template after considering a number of problems with the Normal template.

It would have been tricky to store the code in the Normal template. Every user has their own Normal template. If a user has previously made any default changes, these changes are stored in that user's Normal template. Examples of default changes include:

- Documents will by default have page margins of 1 inch for each of the top, bottom, left and right margins.
- A custom macro that adds signature information to the current document (e.g., Joanna McGrenere, title, address) when Ctrl-M is invoked.
- The **Cut**, **Copy**, and **Paste** toolbar buttons have been removed from the toolbar through the native customize facility (**Tools** menu, **Customize**).

Thus, in order to install the prototype it would not have been possible to simply create a new Normal template containing our code and then use it to replace the user's existing Normal template. If we did this then all default changes made by the user would be lost. Note that if one was to run a laboratory study, replacing the Normal template would be a reasonable thing to do because one wouldn't be concerned about previous changes to the Normal template.

Another possibility would have been to add our code to the user's current Normal template. This would have required copying all the macro code required for the prototype into the user's Normal template[64]. The problem with this approach was that if the prototype had to be uninstalled, the user would have been required to remove the Normal template and restart MSWord. Although removing the Normal template would have the desired effect of uninstalling the prototype it would also have the unintended effect of losing all the user's default settings.

Given these challenges of storing the prototype code in the Normal template we opted instead to create our own global template which we called MultipleInterfaces.dot. In order to delete the prototype one needed only to shut down MSWord, remove the MultipleInterfaces.dot file

[64] To do this one would need to (1) move the user's Normal template from their MSWord User Templates directory, (2) launch MSWord, (3) open their Normal template as if it were a regular document, (4) launch the Visual Basic Editor, (5) copy and paste the macro code into the Normal template, (6) save the Normal template, (7) move it back into the User Templates directory, and lastly (8) re-launch MSWord.

from the Startup directory, and then re-launch MSWord. In order to install a new version of the prototype, one needed only to shut down MSWord, place a new version of the MultipleInterfaces.dot file in the Startup directory, and re-launch MSWord. The user's Normal template was not affected in any way by the MultipleInterfaces.dot template. This mechanism works because MSWord will load any global templates that it finds in the Startup directory, thus ensuring that the prototype is correctly invoked whenever the user launches MSWord.

Creating a new template is easy. For both document and global templates, follow these general steps:

1) **New Blank Document** on the **Standard Toolbar**

2) **File** menu, **Save-As**, put in a file name, **Save-as-type** select Document Template.

3) **Save**

Or

1) **File** menu, **New**…

2) **Create New**, select Template

3) **OK**

Templates have the file extension ".dot". To make a template a global template, place it in the Startup directory. To make the template a document template, place it in either the User Templates or the Workgroup Templates directory. The location of these directories can be found through **Tools** menu, **Options**, **File Locations** tab.

One of the important things that can be maintained in a template is VBA code. Our prototype takes advantage of this and stores all of its VBA code in a global template. This meant that we did not have to modify MSWord (which would have been well beyond the scope of our research) or the users' documents (which would have made it difficult to conduct a field study).

Visual Basic Editor

The Visual Basic editor is the integrated development environment in which one can create, edit, debug, and run code associated with Microsoft Office documents. To open the Visual Basic Editor, click the **Visual Basic Editor** button on the **Visual Basic** toolbar or select **Tools** menu, **Macro**, **Visual Basic Editor**.

Descriptions of the Visual Basic Editor as well as the MSOffice object hierarchy can be found in both of the references given below. An understanding of the MSOffice object hierarchy is necessary because it is the internal data structure that the VBA code must utilize for all of its interactions with MSWord. Fortunately, we found that most of what we needed was documented and suitable (after some trial and error) for our needs. Given that these resources are readily available in the literature, we will not replicate their content here, but will simply refer the reader to them for most of the details of VBA and the MSOffice object hierarchy.

> Writing Word Macros: An Introduction to Programming Word using VBA By Steven Roman, 2nd Edition October 1999
>
> Microsoft Office 97/Visual Basic Programmer's Guide (available through the web on MSDN Library http://msdn.microsoft.com/library/)

In order to add code to a particular document or template, one needs to load the document or template into MSWord and then invoke the Visual Basic Editor. Make sure that the desired document or template is selected in the Project window and then proceed to enter code into the Code window.

VBA Code for Multiple Interfaces Prototype

We created about 5,000 lines of VBA code for our prototype. Our goal was not to build production software, but only to provide a reliable and relatively efficient mechanism for simulating the features we needed.

In this section we describe our implementation approach and we step through some actual VBA code. Our intention is not to provide all of the implementation details, but rather to provide sufficient detail for a programmer to begin to recreate our prototype. Where we do provide actual code, we have in many cases simplified the code by removing extraneous detail.

We explain in the following subsections how the MSOffice menu and toolbar hierarchy is traversed to produce a flat file containing the state of the interface so that the prototype can correctly initialize MSWord each time it is launched, how the MSWord menu is initialized with *Toggle* and *Modify* buttons, how toggling modifies the interface so it appears to switch between the *Full Interface* and the *Personal Interface*, and how personalization through the addition/deletion of functions to/from the personal interface is accomplished.

Storing State Information for the Personal Interface in a Flat File

As mentioned above, template files (both global and document) can maintain state information. Therefore it is possible, for example, to create a new template, reconfigure the toolbar by adding or removing items, and then save that state information to the new template. The next time MSWord launches with the template loaded, the toolbar will be reconfigured in the same way.

We did consider taking advantage of the state-preserving capability of templates for our multiple interfaces prototype, but decided against it for two reasons:

1) If technical problems were experienced and the user had to reinstall the prototype, then the user would be forced to reconstruct his/her personal interface. Reinstalling

the prototype essentially would have meant loss of all personal interface state information.

2) The template mechanism in MSWord is very opaque to the programmer[65]. The programmer can query the state of certain objects in MSWord (e.g., the visibility of a particular menu item) but cannot determine which of the loaded templates is the one that affected the state of any given object or even which objects, if any, have had their state changed. One cannot for example ask the question: what does this template do, i.e., what state changes does it make? The only way this can be done it to load one template at a time and *look* for state changes.

In the end, it was decided that the most straightforward approach to creating the multiple interfaces prototype was to store all the VBA code in the template and to store all the state information, namely the visibility of menu/toolbar items, in a flat file. The flat file would be read in each time MSWord was launched and would be updated according to the user's personalizing activity each time the user used the personalizing mechanism.

The initial flat file was created by traversing the user's **Menu bar**, **Standard Toolbar** and **Formatting Toolbar** (much the same way you would traverse a tree data structure). In this way, if the user had made any customization to his/her menu/toolbars already, this would be captured in the flat file. The createBaseFile subroutine shown below was used to create the initial file.

```
Sub createBaseFile()

    On Error GoTo MyErrorHandler

    Const visible = 0

    'menuIcons is an array of integers that contains the internal
    'indices, or ids, used by MSWord for each function.
    'This array contains only those menu ids that show icons - not all
    'menu items show icons. These ids were obtained manually by
    'traversing the menu hierarchy and outputting all the menu ids and
    'then looking to see which ids show an icon. Only those ids that
    'show an icon appear in this list.
    Const menuIcons =
"18,23,3,3823,109,4,3738,2188,259,938,1707,3251,128,335,21,19,22,141,224,3
621,287,4387,1714,1594,178,1589,767,1576,3260,682,2629,2630,1031,1571,1764
,17,253,779,783,9,782,1706,3623,3743,144,2327,2566,790,1709,2042,3727,4177
,794,186,2780,1695,3631,2059,3392,3685,3687,3681,3683,295,294,293,292,798,
800,107,3907,3908,3909,2068,2067,928,2816,984,124"

    Dim fr As Integer
    Dim cb As CommandBar
```

[65] This statement cannot be stressed enough. A significant amount of time was spent investigating the complexities of the template mechanism.

```
Dim ctl As CommandBarControl
Dim ctl2 As CommandBarControl
Dim ctl3 As CommandBarControl
Dim ctl4 As CommandBarControl
Dim trueBool As Boolean
Dim userName As String

'Items 761, 831, 830, and 3260 are all exceptions that have to
'be handled specially
Dim done761 As Boolean
Dim done831 As Boolean
Dim done830 As Boolean
Dim done3260 As Boolean

Dim error As Boolean
Dim i As Integer
Dim index2 As Integer
Dim index3 As Integer

userName = InputBox("Enter name of user: ", "createBaseFile")

If Not normalBackedUp(userName) Then Exit Sub

directory = Options.DefaultFilePath(Path:=wdStartupPath)

error = False
trueBool = True

fr = FreeFile
Open directory & separator & BASE & "_" & userName & _
  "_" & Format(Date, "mmddyy") & EXTENSION _
  For Output As #fr

'Standard toolbar
Write #fr, "STANDARD", 0, 0, False, 0, 0, False, 0
Set cb = CommandBars("Standard")

For Each ctl In cb.Controls
    Write #fr, ctl.caption, ctl.index, ctl.id, ctl.beginGroup, _
        ctl.Type, ctl.faceId, ctl.visible, visible

    If error Then
        'use id in place of faceId
        Write #fr, ctl.caption, ctl.index, ctl.id, _
              ctl.beginGroup, ctl.Type, ctl.id, ctl.visible, visible
        error = False
    End If
Next

'Formatting toolbar
Write #fr, "FORMATTING", 0, 0, False, 0, 0, False, 0
Set cb = CommandBars("Formatting")

For Each ctl In cb.Controls
    Write #fr, ctl.caption, ctl.index, ctl.id, ctl.beginGroup, _
        ctl.Type, ctl.faceId, ctl.visible, visible
    If error Then
        'use id in place of faceId
        Write #fr, ctl.caption, ctl.index, ctl.id, ctl.beginGroup, _
              ctl.Type, ctl.id, ctl.visible, visible
        error = False
    End If
Next

'Menu
Write #fr, "MENU", 0, 0, False, 0, 0, False, 0
Set cb = CommandBars("Menu bar")

done761 = False
done831 = False
done830 = False
done3260 = False

'main menus -- File, Edit, View, etc.
For Each ctl In cb.Controls

    Write #fr, ctl.caption, ctl.index, ctl.id, ctl.beginGroup, _
        ctl.Type, 0, ctl.visible, visible

    If ctl.Type = msoControlPopup Then

        index2 = 0

        'first-level menu items
        For Each ctl2 In ctl.Controls

            index2 = index2 + 1

            If ctl2.Type = msoControlPopup Then

                Write #fr, ctl2.caption, index2, ctl2.id, _
                    ctl2.beginGroup, ctl2.Type, 0, _
                    ctl2.visible, visible

                index3 = 0

                'second-level menu items
                For Each ctl3 In ctl2.Controls

                    index3 = index3 + 1

                    'should the control show an icon?
                    'this is really a bit of a hack
                    i = InStr(1, menuIcons, CStr(ctl3.id), _
                          vbTextCompare)

                    If (ctl3.id = 3260) Then
                        If Not (done3260 = True) Then
                            Write #fr, "<list of autotext _
                                entries>", index3, ctl3.id, _
                                ctl3.beginGroup, ctl3.Type, 0, _
                                ctl3.visible, visible
                            done3260 = True
                        Else
                            index3 = index3 - 1
                        End If

                    ElseIf (ctl3.id = 761) Then
                        If Not (done761 = True) Then
                            Write #fr, "<list of toolbars>", _
                                index3, ctl3.id, ctl3.beginGroup, _
                                ctl3.Type, 0, ctl3.visible, visible
                            done761 = True
                        Else
                            index3 = index3 - 1
                        End If

                    ElseIf i = 0 Then
                        Write #fr, ctl3.caption, index3, ctl3.id, _
                          ctl3.beginGroup, ctl3.Type, 0, _
                          ctl3.visible, visible
                    Else
                        Write #fr, ctl3.caption, index3, ctl3.id, _
                          ctl3.beginGroup, ctl3.Type, ctl3.faceId, _
                          ctl3.visible, visible
                    End If

                    If error Then
                        Write #fr, ctl3.caption, index3, ctl3.id, _
                          ctl3.beginGroup, ctl3.Type, ctl3.faceId, _
                          ctl3.visible, visible
                        error = False
                    End If
                Next
                'The current construction of the personal menu does
                'not require these end of menu notations.
                Write #fr, "EndOfSubMenu", 0, 0, False, 0, 0, False, 0


            Else 'not popup

                i = InStr(1, menuIcons, CStr(ctl2.id), vbTextCompare)

                If (ctl2.id = 830) Then
                    If Not (done830 = True) Then
                        Write #fr, "<list of active windows>", _
                          index2, ctl2.id, ctl2.beginGroup, _
                          ctl2.Type, 0, ctl2.visible, visible
                        done830 = True
                    Else
                        index2 = index2 - 1
                    End If

                ElseIf (ctl2.id = 831) Then
                    If Not (done831 = True) Then
                        Write #fr, "<list of recent files>", _
                          index2, ctl2.id, ctl2.beginGroup, _
                          ctl2.Type, 0, ctl2.visible, visible
                        done831 = True
                    Else
```

```
                            index2 = index2 - 1
                        End If

                    ElseIf i = 0 Then
                        Write #fr, ctl2.caption, index2, ctl2.id, _
                              ctl2.beginGroup, ctl2.Type, 0, _
                              ctl2.visible, visible
                    Else
                        Write #fr, ctl2.caption, index2, ctl2.id, _
                              ctl2.beginGroup, ctl2.Type, ctl2.faceId, _
                              ctl2.visible, visible
                    End If

                    If error Then
                        Write #fr, ctl2.caption, index2, ctl2.id, _
                              ctl2.beginGroup, ctl2.Type, 0, _
                              ctl2.visible, visible
                        error = False
                    End If
                End If

            Next
            Write #fr, "EndOfMenu", 0, 0, False, 0, 0, False, 0
        End If
    Next
    Write #fr, "END", 0, 0, False, 0, 0, False, 0

    Close #fr
    Exit Sub

MyErrorHandler:
    Debug.Print "createBaseFile "; Err.Description
    error = True
    Resume Next
End Sub
```

The record format for the flat file was as follows:

caption: text that is displayed in menu (for menu items) or in tooltip (for toolbar items) – the ampersand (&) shown in the caption is used to determine the accelerator key for the menu item (e.g., press the keys **Alt**, **f**, **a** in sequence for **File** menu, **Save As**)

index: index into toolbar/menu structure

control id: MSWord’s identifier for this item

begin group: does item begin a group, i.e., have a separator bar in front of it?

control type: MSWord’s control type, e.g., popup menu

icon identifier: MSWord's identifier for the icon associated with the item – set to 0 if it is not visible

default visibility: is this item visible by default (independent of the prototype)?

item visibility: set to 1 if it will be visible in the interface defined by the current flat file

Note that the **begin group**, **control type**, **icon identifier**, and **default visibility** fields were added for the Study Two version of the prototype. Those fields were used to create the "fake" menu/toolbars used in the personalizing mechanism as described in the Section *Personalizing Mechanism* below.

A sample flat file ("`Joanna's Interface.int`") is shown below.

```
"STANDARD",0,0,#FALSE#,0,0,#FALSE#,0
"New &Blank Document",1,2520,#FALSE#,1,3813,#TRUE#,1
"&Open...",2,23,#FALSE#,1,23,#TRUE#,1
"&Save",3,3,#FALSE#,1,3,#TRUE#,0
"&Mail Recipient",4,3738,#FALSE#,1,3738,#FALSE#,0
"&Print",5,2521,#TRUE#,1,2521,#TRUE#,1
"Print Pre&view",6,109,#FALSE#,1,109,#TRUE#,1
"&Spelling and Grammar...",7,2566,#FALSE#,1,2566,#TRUE#,1
"Cu&t",8,21,#TRUE#,1,21,#TRUE#,0
"&Copy",9,19,#FALSE#,1,19,#TRUE#,0
"&Paste",10,22,#FALSE#,1,22,#TRUE#,0
"&Format Painter",11,108,#FALSE#,1,108,#TRUE#,1
"Can't &Undo",12,128,#TRUE#,6,128,#TRUE#,1
"Can't &Redo",13,129,#FALSE#,6,129,#TRUE#,1
"Hyperl&ink...",14,1576,#TRUE#,1,1576,#TRUE#,0
"&Tables and Borders Toolbar",15,916,#FALSE#,1,916,#TRUE#,0
"&Insert Table...",16,333,#FALSE#,16,333,#TRUE#,1
"&Insert Excel Spreadsheet",17,142,#FALSE#,16,142,#TRUE#,0
"&Columns...",18,9,#FALSE#,16,9,#TRUE#,1
"&Drawing",19,204,#FALSE#,1,204,#TRUE#,1
"&Document Map",20,1714,#TRUE#,1,1714,#TRUE#,1
"&Show All",21,119,#FALSE#,1,119,#TRUE#,1
"&Zoom:",22,1733,#FALSE#,4,1733,#TRUE#,1
"Microsoft Word &Help",23,984,#FALSE#,1,984,#TRUE#,0
"FORMATTING",0,0,#FALSE#,0,0,#FALSE#,0
"&Style:",1,1732,#FALSE#,20,1732,#TRUE#,1
"&Font:",2,1728,#FALSE#,4,1728,#TRUE#,1
"&Font Size:",3,1731,#FALSE#,4,1731,#TRUE#,1
"&Keyboard Language",4,3659,#TRUE#,13,3659,#FALSE#,0
"&Bold",5,113,#TRUE#,1,113,#TRUE#,0
"&Italic",6,114,#FALSE#,1,114,#TRUE#,0
"&Underline",7,115,#FALSE#,1,115,#TRUE#,0
"Align &Left",8,120,#TRUE#,1,120,#TRUE#,1
"&Center",9,122,#FALSE#,1,122,#TRUE#,1
"Align &Right",10,121,#FALSE#,1,121,#TRUE#,1
"&Justify",11,123,#FALSE#,1,123,#TRUE#,1
"&Left-to-Right",12,1846,#TRUE#,1,1846,#FALSE#,0
```

```
"&Right-to-Left",13,1847,#FALSE#,1,1847,#FALSE#,0
"&Numbering",14,11,#TRUE#,1,11,#TRUE#,1
"&Bullets",15,12,#FALSE#,1,12,#TRUE#,1
"&Decrease Indent",16,3473,#FALSE#,1,14,#TRUE#,1
"&Increase Indent",17,3472,#FALSE#,1,15,#TRUE#,1
"&Borders",18,203,#TRUE#,14,203,#TRUE#,1
"&Highlight",19,340,#FALSE#,13,340,#TRUE#,0
"&Font Color",20,401,#FALSE#,13,401,#TRUE#,0
"MENU",0,0,#FALSE#,0,0,#FALSE#,0
"&File",1,30002,#FALSE#,10,0,#TRUE#,1
"&New...",1,18,#FALSE#,1,18,#TRUE#,1
"&Open...",2,23,#FALSE#,1,23,#TRUE#,0
"&Close",3,106,#FALSE#,1,0,#TRUE#,0
"&Save",4,3,#TRUE#,1,3,#TRUE#,0
"Save &As...",5,748,#FALSE#,1,0,#TRUE#,1
"Save as Web Pa&ge...",6,3823,#FALSE#,1,3823,#TRUE#,0
"V&ersions...",7,2522,#FALSE#,1,0,#TRUE#,0
"We&b Page Preview",8,3655,#TRUE#,1,0,#TRUE#,0
"Page Set&up...",9,247,#TRUE#,1,0,#TRUE#,1
"Print Pre&view",10,109,#FALSE#,1,109,#TRUE#,1
"&Print...",11,4,#FALSE#,1,4,#TRUE#,1
"C&reate Adobe PDF...",12,1,#FALSE#,1,1,#TRUE#,0
"Sen&d To",13,30095,#TRUE#,10,0,#TRUE#,0
"&Mail Recipient",1,3738,#FALSE#,1,3738,#FALSE#,0
"M&ail Recipient (as Attachment)...",2,2188,#FALSE#,1,2188,#TRUE#,0
"&Routing Recipient...",3,259,#FALSE#,1,259,#TRUE#,0
"&Exchange Folder...",4,938,#FALSE#,1,938,#TRUE#,0
"&Online Meeting Participant",5,3728,#FALSE#,1,0,#TRUE#,0
"&Fax Recipient...",6,1707,#FALSE#,1,1707,#TRUE#,0
"Microsoft &PowerPoint",7,3251,#TRUE#,1,285,#TRUE#,0
"EndOfSubMenu",0,0,#FALSE#,0,0,#FALSE#,0
"Propert&ies",14,750,#FALSE#,1,0,#TRUE#,0
"<list of recent files>",15,831,#TRUE#,1,0,#TRUE#,0
"E&xit",16,752,#TRUE#,1,0,#TRUE#,0
"EndOfMenu",0,0,#FALSE#,0,0,#FALSE#,0
...
```

<Edit menu through Help menu are omitted for brevity.>

```
...
"END",0,0,#FALSE#,0,0,#FALSE#,0
```

Note that the prototype could handle more than one interface (in addition to the full interface). An interface was defined simply by placing a ".int" file in the Startup directory. We chose that suffix because it was available. This could easily be changed to some other suffix. More than one file could be placed in the directory and used by the prototype. All the "*.int" files for a given user were identical in content except for the values in the visibility field. The filenames became the interface names shown in the toggle, e.g., "Joanna's Interface.int" and "Minimal Interface.int" were shown as Joanna's Interface and Minimal Interface in the toggle.

Initialization – Adding the Toggle and Modify Button

Each time MSWord was launched, our prototype needed to "take over". This was accomplished by writing VBA code that read the flat file and set up the menus and toolbars accordingly, and added the toggling and personalizing buttons. In general, any code that needs to be executed automatically when MSWord is launched must be put in a subroutine named AutoExec in the Main module.

In our AutoExec routine we set the customization context to be our MultipleInterfaces.dot template. This meant that any customizations that were carried out were stored to our template rather than the Normal template. As we have mentioned, no state information should ever have been stored in the template during normal operation. The only time this did occur was when the code did not terminate properly as in the case of debugging the prototype[66].

Our AutoExec subroutine is shown below. Following that we show the code for our AddToggle and addModifyButton subroutines. Note that the AutoExec ran each time MSWord was launched – thus the toggle and modify button were added each time.

```
Sub AutoExec()
    Dim result As Boolean

    Set MItemplate = getTemplate(MULTIPLE_INTERFACE_TEMPLATE)
    CustomizationContext = MItemplate

    GetInterfaceNames  'looks for .int files

    'if lastInterface is 1 then all we have is the default interface
    If Not (lastInterface = 1) Then

        noProtectCommandBars
        AddToggle

        CommandBars.AdaptiveMenus = False
        protectCommandBars
        disableNativeCustomize  'makes native customize inaccessible
        setupPersonalInterface
        Customize.addModifyButton

    End If
```

[66] Throughout the process of creating/editing/debugging the MultipleInterfaces.dot template state information would sometimes get saved into the template. One way to counteract this was to write a reset macro that essentially restored the toolbars/menu (e.g., CommandBars("Menu Bar").reset) and deleted any toolbars that the prototype created (e.g., CommandBars(Customize.ADD_MENUBAR_NAME).Delete). This reset macro could be called prior to saving the MultipleInterfaces.dot template. The goal was to save any changes made to the macro code, but not save any state changes to the template. To fully counteract saving state changes, every once in a while we would create a completely new template, cut and paste all our macro code into the new template, and rename it MultipleInterfaces.dot.

```
End Sub
```

```
Sub AddToggle()
    Dim cb As CommandBar
    Dim ctl As CommandBarControl
    Dim cbl As CommandBarControl
    Dim i As Integer

    CustomizationContext = MItemplate

    'Check to make sure toggle isn't there already
    Set cbl = CommandBars("Menu Bar").FindControl(Tag:=TOGGLE_TAG)
    Set cb = CommandBars("Menu bar")

    If cbl Is Nothing Then
        Set cbl = cb.Controls.Add(Type:=msoControlDropdown)
        With cbl
            .Tag = TOGGLE_TAG
            .visible = True
            .caption = TOGGLE_CAPTION
            .DropDownLines = 0
            .DropDownWidth = 120
            .Width = 120
        End With

        cbl.OnAction = "Toggle"  'our toggling subroutine
    Else
        'need to clear it in case additional interfaces have been added
        cbl.Clear
    End If

    For i = 1 To (lastInterface)
        cbl.AddItem interfaces(i), i
    Next i
    cbl.ListIndex = currentInterface

    'adding the toggle sets a dirty bit so make Word think the
    'template has been saved. If you don’t do this then the user will
    'be prompted to save the MultipleInterfaces.dot file when he/she
    'exits MSWord
    MItemplate.Saved = True

End Sub
```

```
Sub addModifyButton()
'Add the modify button beside the toggle

    Dim cb As CommandBar
    Dim ctl As CommandBarControl

    CustomizationContext = Main.MItemplate

    Set cb = CommandBars("Menu bar")
    Set ctl = cb.FindControl(Tag:=UPDATE_TAG)
    If ctl Is Nothing Then
        noProtectCommandBars
        Set ctl = cb.Controls.Add(Type:=msoControlButton)

        With ctl
            .Tag = UPDATE_TAG
            .visible = True
            .caption = "Modify " & Main.PIcaption
            .DescriptionText = "Modify Personalized Interface"
            .OnAction = "update"
            '.Style = msoButtonIcon
            .Style = msoButtonIconAndCaption
            .Width = 120
            .faceId = 352
        End With
        protectCommandBars
    End If

    Main.MItemplate.Saved = True
End Sub
```

Toggling Between Interfaces

Toggling between the interfaces involved setting the visibility of the menu/toolbar items. There was only one set of menu/toolbars that were used regardless of the interface displayed – it was the visibility of the items within the menu/toolbars that distinguished one interface from another.

The code to display the personalized interface (or any interface specified by a “.int” file) simply read in the flat file and updated the menu/toolbars according to the visibility information in that flat file. Below we show the portion of the setupPersonalInterface subroutine that updates the **Standard Toolbar**.

If the interface selected was the full interface, then the toolbars and menu were restored to show all items that would be available in an out-of-the-box version of MSWord. The code for this is not shown but simply required traversing through the toolbar/menu structure and setting the visibility parameter of all of the items to true. This is very similar to the code used to create the flat file.

Toggling between interfaces also generated a record to our toggle log file. These records were later merged with the IV logger records (described in Section 6.2.4) to provide a full record of the user’s activities.

```
Sub Toggle()

    Dim cbl As CommandBarControl
    Dim change As Boolean
    On Error GoTo MyErrorHandler

    CustomizationContext = MItemplate

    Set cbl = CommandBars("Menu Bar").FindControl(Tag:=TOGGLE_TAG)

    If Not cbl Is Nothing Then

        change = Not (currentInterface = cbl.ListIndex)

        If (change) Then

            currentInterface = cbl.ListIndex

            If currentInterface = lastInterface Then
                restoreDefaultTBS
                restoreDefaultMenu
            Else
                setupPersonalInterface
            End If

            'Add a record to Toggle file
            Dim Filename As String
            Dim datestamp As String
            Dim timestamp As String
            Dim fr As Integer
            Dim mydate, mytime

            mydate = Date
            mytime = Time
            datestamp = Format(mydate, "mm/dd/yy")
            timestamp = Format(mytime, "hh:mm:ss")

            fr = FreeFile
            Open (directory & separator & TOGGLE_FILE) For Append As #fr
            Print #fr, datestamp; " "; timestamp; Chr(9); cbl.ListIndex
            Close #fr
        End If

        'I'm not sure why this needs to be set again, but for some
        'reason if ListIndex is not set again, it sometimes unsets
        'itself, for example, after opening and closing a new document.
        cbl.ListIndex = currentInterface

    End If

    MItemplate.Saved = True
    Exit Sub

MyErrorHandler:
    Debug.Print "Toggle"; Err.Description
    Resume Next

End Sub
```

Excerpt of setupPersonalInterface()

```
'This routine assumes a properly constructed input file. If the
'file has been tampered with it may not work.
fr = FreeFile
Open (directory & separator & interfaces(currentInterface) & _
    EXTENSION) For Input As #fr

'read until you get to STANDARD in case there is any blank lines
'at the beginning
Do Until (EOF(fr) Or (caption = "STANDARD"))
    Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
         defVisible, visible
Loop


'update the standard toolbar -- end is signaled by the
'FORMATTING tag
Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
    defVisible, visible
Do Until (EOF(fr) Or (caption = "FORMATTING"))
    Set ctl = CommandBars("Standard").Controls(index)
    ctl.visible = visible
    Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
        defVisible, visible
Loop
```

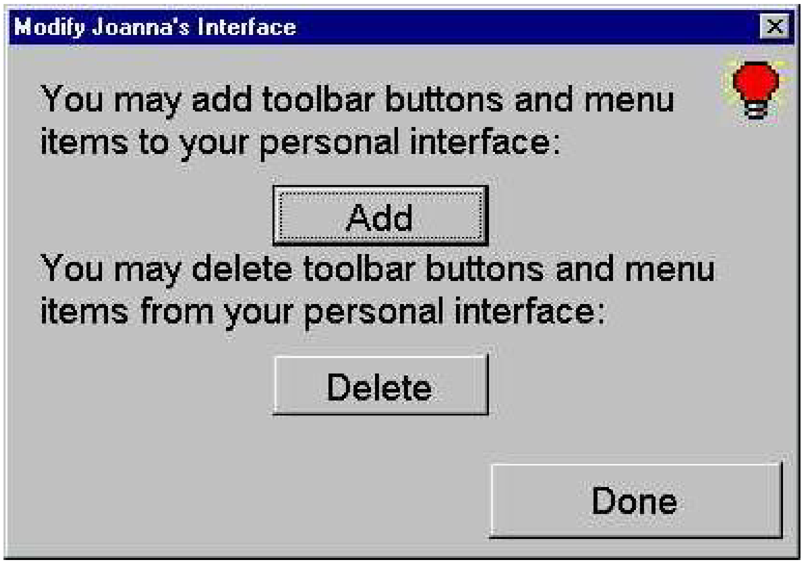
Note that an alternate way that one might first approach having a personalized interface would be to have separate toolbars/menu for the personalized interface than from those of the full interface. Toggling between the interfaces would simply require hiding one set of toolbars/menu and displaying another. We initially tried this approach but the problem was that certain items on the toolbars/menu, namely automation items, cannot be copied. Therefore we could not fully replicate the real toolbars/menu. Similar anomalies with these commands also affected the design of the personalizing mechanism, which is described next.

Personalizing Mechanism

For the version of the prototype that we used in our Pilot Study there was no mechanism for users to personalize their own interfaces. Rather the researcher constructed the personalized interfaces by setting the visibility information directly in the flat files.

For Study Two we updated the prototype to include a personalizing mechanism. This mechanism was achieved through the addition of a modify button to the **Menu bar**, a series of forms, and a number of subroutines. The modify button was described above in the Section *Initialization – Adding the Toggle and Modify Button*, and here we describe the forms and the subroutines. We step through portions of the code that were executed when the user added items to his/her personalized interface.

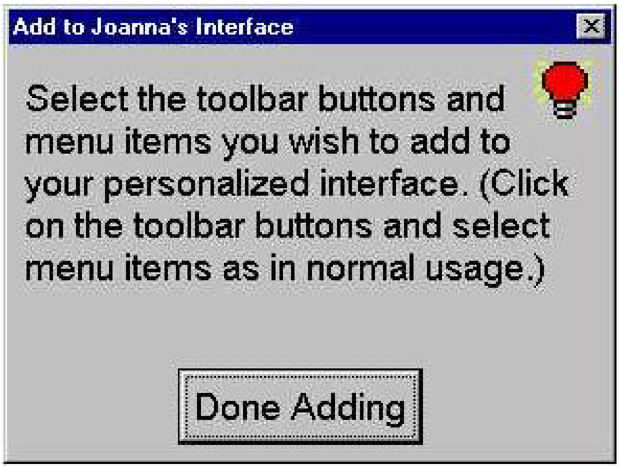
The forms were created in the Visual Basic Editor through a direct manipulation interface, such that each control (e.g., a button) was added manually to an empty form. The “Modify Personalized Interface” form (shown below) was displayed when the user clicked on the modify button on the menu. Each form or control was given a name – the name is simply used within VBA to refer to the object. We gave this form the name UpdateGeneral.



You may add toolbar buttons and menu items to your personal interface:

You may delete toolbar buttons and menu items from your personal interface:

The Add button shown in the middle of the form was given the name AddButton. By creating a subroutine with the name AddButton_Click associated with the UpdateGeneral form we were able to define what would be executed when the button was clicked on. In this routine we recorded the saved state of the current document, i.e., has it been saved since the last modification was made? We then hid the "Modify Personalized Interface" form and showed the "Add to Personalized Interface" form. We also called the doAdd subroutine which is described below.



Select the toolbar buttons and menu items you wish to add to your personalized interface. (Click on the toolbar buttons and select menu items as in normal usage.)

```
Private Sub AddButton_Click()

    If Not Documents.Count = 0 Then
        activeDocPriorState = ActiveDocument.Saved
    End If
    UpdateGeneral.Hide

    UpdateAdd.Show
    Customize.doAdd
End Sub
```

The doAdd subroutine created the infrastructure for the user to select items to add to his/her personalized interface. The basic idea was that we created almost identical copies of the user's **Menu bar**, **Standard Toolbar**, and **Formatting Toolbar**. These were created based on the information stored in the ".int" file that defined the interface being personalized. The user was able to select toolbar/menu items from these "fake" toolbars/menu. Instead of the items on these fake toolbars/menu actually invoking the associated function, they were set to add the id of the item selected to a list. There were actually three lists – one for each of the two toolbars and one for the menu. The reason that separate lists were maintained was that toolbar/menu item ids are not unique (e.g., the control id for Cut is 21 in both the **Standard Toolbar** and in the **Edit** menu). These lists were then used to update the real toolbars/menu when the user exited from the personalizing mode.

One of the limitations of VBA is that it will not accept a parameterized subroutine (i.e. a function) as the action that will be carried out when a toolbar/menu item is selected. So we weren't able to simply use one general subroutine that was passed the item id as a parameter for all the fake toolbar/menu items. Instead we had to create an individual wrapper subroutine for each individual item on the fake toolbars/menu. The line in the code that did this is shown below:

```
        contl.OnAction = "Add.standard_" + CStr(id)
```

So, for example, the **New Blank Document** toolbar button has the id 2520 and so the expression resolves to:

```
        contl.OnAction = "Add.standard_2520"
```

The standard_2520 subroutine is shown after doAdd().

```
Sub doAdd()
'prepare for user to select items to add

    Dim cb As CommandBar

    CustomizationContext = Main.MItemplate

    On Error GoTo MyErrorHandler

    'lock document editing during update so that user can only do
    'the update
    If Not Documents.Count = 0 Then
        If ActiveDocument.ProtectionType = wdNoProtection Then
            ActiveDocument.Protect (wdAllowOnlyComments)
        End If
    End If

    'setting the menu bar to invisible is not needed because there can
    'only be one menu bar
    'CommandBars("Menu bar").visible = False
    CommandBars("Standard").visible = False
    CommandBars("Formatting").visible = False

    'lists (global variables) that will contain items user has added to
    'each of the toolbars and menu
    Set addStandardCol = Nothing
    Set addFormattingCol = Nothing
    Set addMenuCol = Nothing

    'create fake menu and toolbars
    Set addMenuBar = CommandBars.Add(ADD_MENUBAR_NAME, msoBarTop, _
        True, True)
    Set addStandardTB = CommandBars.Add(ADD_STANDARDTB_NAME, _
        msoBarTop, False, True)
    Set addFormattingTB = CommandBars.Add(ADD_FORMATTINGTB_NAME, _
        msoBarTop, False, True)

    action = "Add "

    'use the information in the personalized interface flat file to
    'populate the menu/toolbars with items
    fr = FreeFile
    Open (Main.directory & Main.separator & Main.PIfilename) For _
        Input As #fr

    'read until you get to STANDARD in case there is any blank lines at
    'the beginning
    Do Until (EOF(fr) Or (caption = "STANDARD"))
        Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
            defVisible, visible
    Loop


    'update the standard toolbar -- end is signaled by the
    'FORMATTING tag
    Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
        defVisible, visible
    Do Until (EOF(fr) Or (caption = "FORMATTING"))

      If Not (ctlType = 4) And Not (ctlType = 20) Then
       Set contl = addStandardTB.Controls.Add(Type:=msoControlButton)
       contl.faceId = faceId
      Else
       Set contl = addStandardTB.Controls.Add(Type:=msoControlComboBox)
       contl.Text = removeAmp(caption) 'remove ampersand from caption
       contl.AddItem action, 1
      End If

        contl.caption = action & caption
        contl.beginGroup = beginGroup
        contl.OnAction = "Add.standard_" + CStr(id)
```

```
        contl.visible = defVisible

        'if the item is already available in the user’s personal
        'interface, then gray it out
        If visible Then
            contl.Enabled = False
        End If

        Input #fr, caption, index, id, beginGroup, ctlType, faceId, _
            defVisible, visible
    Loop
```

<Populating the addFormatting toolbar and the addMenuBar with toolbar buttons and menu items is omitted for brevity. >

```
    'now make them visible
    CommandBars(ADD_STANDARDTB_NAME).visible = True
    CommandBars(ADD_FORMATTINGTB_NAME).visible = True
    CommandBars(ADD_MENUBAR_NAME).visible = True

    Close #fr

    Main.MItemplate.Saved = True
    Exit Sub

MyErrorHandler:
    Debug.Print "doAdd: " & Err.Description
    Resume Next

End Sub
```

```
Sub standard_2520()
    Dim msg As String

    msg = "New Blank Document"
    Customize.StandardAdd 2520, msg, CommandBars.ActionControl
End Sub
```

```
Sub StandardAdd(id As Integer, caption As String, ctl As _
                CommandBarControl)

    If addStandardCol Is Nothing Then
        Set addStandardCol = New Collection
    End If

    addStandardCol.Add id

    'show the confirmation dialog box
    Added.FunctionCaption = caption
    Added.Show

    If Not ctl Is Nothing Then
        ctl.Enabled = False
    End If
End Sub
```

Once the doAdd subroutine had created the fake toolbars/menu, the user could select items as in normal usage. For each item that was selected there was a confirmation dialog box displayed. The end of personalizing was signaled when the user selected the Done Adding button on the "Add to Personalized Interface" form. This form remained visible until Done Adding was selected.

The following code was executed when the Done Adding button was selected:

```
Private Sub addingDone_Click()
    UpdateAdd.Hide
    Customize.doneAdd
    UpdateGeneral.Show
End Sub
```

The doneAdd subroutine checked to see if anything had been added (were the addition lists empty?), and if not it called the updatePIfileAdd subroutine. It then deleted the fake toolbars/menu and restored the real ones. The updatePIfileAdd subroutine is not shown here but it updated the ".int" file based on the content of the addition lists. It also output the content of the addition lists to a log file that captured the user's personalizing activity. This log was then merged with the IV logger records as described in Section 6.2.4. The personal interface was then displayed as described in the Section *Toggling Between Interfaces* above.

```
Sub doneAdd()
'called once the user has finished selecting items to be added to his/her
personalized interface

    CustomizationContext = Main.MItemplate

    If Not ((addStandardCol Is Nothing) And (addFormattingCol Is Nothing)
And (addMenuCol Is Nothing)) Then
        updatePIfileAdd
    End If

    Main.forceToggle Main.iniPI

    CommandBars(ADD_STANDARDTB_NAME).Delete
    CommandBars(ADD_FORMATTINGTB_NAME).Delete
    CommandBars(ADD_MENUBAR_NAME).Delete

    CommandBars("Menu bar").visible = True
    CommandBars("Standard").visible = True
    CommandBars("Formatting").visible = True

    If Not Documents.Count = 0 Then
        If Not ActiveDocument.ProtectionType = wdNoProtection Then
            ActiveDocument.Unprotect
            'protecting and unprotecting the document will cause the
            'saved attribute to be set to false
            'restore the saved state from before altering the interface
            ActiveDocument.Saved = UpdateGeneral.activeDocPriorState
        End If
    End If

    Main.MItemplate.Saved = True

End Sub
```

Notes:

Each time the user entered the personalizing mode the fake toolbars/menu were created and then destroyed. The reason for this was that our fake toolbars were actual toolbars just like any other in MSWord (e.g., **Drawing**, **Forms**, **Picture**). If we had not destroyed these toolbars upon exiting the personalizing mode, then the user would have been able to access them just as the other toolbars were accessed (e.g., through **View** menu, **Toolbars**). We could not find a workaround to this and so we created/destroyed the fake toolbars/menu each time the user personalized. For similar reasons, the real toolbars/menu were made invisible and then made visible again after personalizing was complete.

An alternate way that one might first think to approach the task of personalizing would be to have the items of the real toolbars/menu retargeted while in the personalizing mode. The items **onAction** parameters could be set just as we have set them in the fake toolbars/menu, namely set to add the item's identifier to a change list rather than to invoke the real function. When personalizing is complete, the **onAction** parameters would be reset to the original function invocation. The problem with this is that there are many toolbar/menu items that do not allow you to change their **onAction** parameter. A runtime automation error is generated.