Accepted for publication in The Proceedings of Graphics Interface 2000, Montreal, May 2000.

Are We All In the Same "Bloat"?

Joanna McGrenere
Department of Computer Science
University of Toronto
Toronto, ON Canada, M5S 3G4
joanna@dgp.utoronto.ca

Gale Moore
Knowledge Media Design Institute
University of Toronto
Toronto, ON Canada, M5S 2Z9
gmoore@dgp.utoronto.ca

Abstract

"Bloat", a term that has existed in the technical community for many years, has recently received attention in the popular press. The term has a negative connotation implying that human, or system performance is diminished in some way when "bloat" exists. Yet "bloat" is seldom clearly defined and is often a catch-all phrase to suggest that software is filled with unnecessary features. However, to date there are no studies that explore how users actually *experience* complex functionality-filled software applications and most importantly, the extent to which they experience them in similar/different ways. The significance of understanding users' experience is in the implications this understanding has for design. Using both quantitative and qualitative methods, we carried out a study to gain a better understanding of the experiences of 53 members of the general population who use a popular word processor, Microsoft Word, Office 97. As a result we are able to further specify the term "bloat", distinguishing an objective and subjective dimension. It is the discovery of the subjective dimension that opens the design space and raises new challenges for interface designers. There is certainly more to "bloat" than meets the eye.

Keywords: Complex software, bloat, creeping featurism, user experience, office applications, human-centred design, user study, evaluation, personalization.

1 Introduction

Over the past two decades desktop computing has become an integral part of the experience of work across economic sectors and occupations in the advanced economies. The spreadsheet was the first "killer app" for personal computers, followed shortly by the word processor, and the office suites of the late 1980s. These applications competed within their class in the marketplace in terms of the number of functions offered – a phenomenon that became known as the *Feature War*. The assumption was that the greater the number of features, the more useful, or at least the more marketable the application. These applications became increasingly complex in a number of ways. Not only were there more options available, but some of the options offered were sophisticated and required a more complete understanding of computers and traditional printing and publishing practices. Furthermore, the interface itself had become visually more complex, e.g., menus and submenus were growing longer. Simultaneously, there was an explosion in the size and diversity of the user population, many of whom were unfamiliar with either computers or printing or both. Technical developments such as the GUI, and attention to usability have led to improvements over the years, but the impact of this functionality explosion on the actual experiences of those using the tools has received little attention in the literature. However, we are encouraged by the National Academy of Sciences announcement that they are in the early stages of developing an agenda on what they call the "every-citizen" interface [9].

In the past few years, there has been considerable interest in the popular press and the computer world in what has been termed "bloat" or "bloatware" [7] and "creeping featurism"[10]. "Bloat" is a term that has been used 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 the complexity of use" [11]. Creeping featurism, on the other hand, is the tendency to complicate a system by adding features in an ad-hoc, non-systematic manner [11]. One implication is that a bloated application is one in which there are a large number of unused features. In the popular press, "bloat" is often used as a catch-all phrase to suggest, negatively, that an application is filled with unnecessary features [1,3].

But, do users actually *experience* complex software applications in this way? (We define complex software as software with many features, most applications packages today.) We initially assumed that the "average" user must be struggling with applications such as word processors and spreadsheets. But, is this

the case? And if it is, is this primarily related to unused functionality? Do people feel overwhelmed by the number and variety of choices in the interface, and if they do, how do they handle this? A goal of this paper is to specify more closely how users actually experience this complexity, and how they describe it. 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 these data 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 [7] which distinguished two profiles of users according to their perception of "bloat". We thus have several distinct methodological approaches, each offering a unique perspective on the problem. By triangulation of methods we are able to more fully understand the user's experience of complex software applications, and to gain insights that can be applied to interface design.

2 Previous work

To date, there has been no systematic study of how users experience complex software. Some early work has been done on logging the use of functions in UNIX [5, 6, 13]. This was followed by Carroll and Carrithers' now classic work on the "Training Wheels" interface for an early word processor. In this work they found 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 [2]. Greenberg's work on Workbench for UNIX offered another solution. He labeled systems in which users use only a small subset of the command repertory as *recurrent systems* and created a reuse facility; this is a front end that collects the user's commands and then makes them easily accessible for reuse [5]. Linton's [8] recent work on word processors proposes a *recommender system* [12] that alerts users to functionality currently being used by co-workers doing similar tasks. All these researchers with the exception of Carroll and Carrithers have used software logging to capture their subjects' use of commands in the context of carrying out their everyday tasks. Results of these studies consistently show that users of complex software use very few of the commands available the majority of the time; informally this is called the 80/20 rule.

3 Study Design

3.1 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 was selected (MSWord). We used the number of functions available in an application as our indicator of complexity[1]. Because our primary question had to do with the users' 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.

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

- Each *final* menu item in the menus from the menu toolbar counts as 1. A menu item is not considered final if it results in a cascading menu.
- Each item on a toolbar counts as 1 (even if it provides a drop down menu – e.g., borders, styles, and font colour).
- Each button on a scrollbar counts as 1 except the scroll widget (which is composed of scroll left/up and scroll right/up buttons as well as the bar itself) which counts as 1.
- The selectable items on the status bar each count as 1.

Using these heuristics we counted 265 functions in the default MSWord interface. The second-level count included all options available on the first-level dialog boxes[2]–709 options available[3]. In this paper we report only on the first-level functions (n= 265).

3.2 Sample

The sample consisted of 53 participants selected from the general population. While this was not a simple random sample (and therefore it is not appropriate to use inferential statistics), participants were selected with attention to achieving as representative a sample of

[1] There are clearly other factors contributing to the 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 paper we focus specifically on the number of functions.

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

[3] Space limitations do not permit us to list the detailed heuristics used to count the dialog-box functions; these heuristics are considerably more complex than those of the main interface and are available from the authors. By comparison Gibbs reported that there were a total of 1033 functions in MSWord 97, but gave no source. Therefore, we do not know the method used to count the functions [4].

the general adult population as possible. That is, we paid particular attention to achieving representation in terms of variables such as age, gender, education, occupation and organizational status.

3.3 Instruments and Data collection

Functionality Interview
A researcher presented each participant with two series of printed screen captures. 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 two-point scale: *familiar* and *unfamiliar*. Responses to question two were scored on a three-point scale: *used regularly*, *used irregularly*, and *not used*. 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.

We were particularly concerned with ecological validity, and 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 and to provide a measure of reliability, participants were told at the outset that they would be asked periodically to describe the action of any function with which they 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.

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 in general, and MSWord, in particular.

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

Questionnaire
Prior to the interview each participant was given a poll-type questionnaire. 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 provided.

3.4 How our work is differentiated from others

Our work can be differentiated from earlier research in a number of ways. First, we use multiple methods and the results show this provides us with a unique perspective for thinking about complex software. The different approaches do not provide conflicting answers, but rather help to elaborate an extremely complex question, each highlighting a specific dimension of the problem. Quantitative methods offer a detailed description of the feature space, questionnaire responses helped identify patterns and summarise 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 are 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 self-reported data are subject to the participant's 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 compensate for the limitation of each method used on its own.

The next three sections present our findings and analysis.

4 Quantitative method: Software is "bloated" when a significant proportion of the functions available are not used by the majority of users

By this definition, MSWord is "bloated" indeed, as the pie charts illustrate (Figure 1). Compare the number of

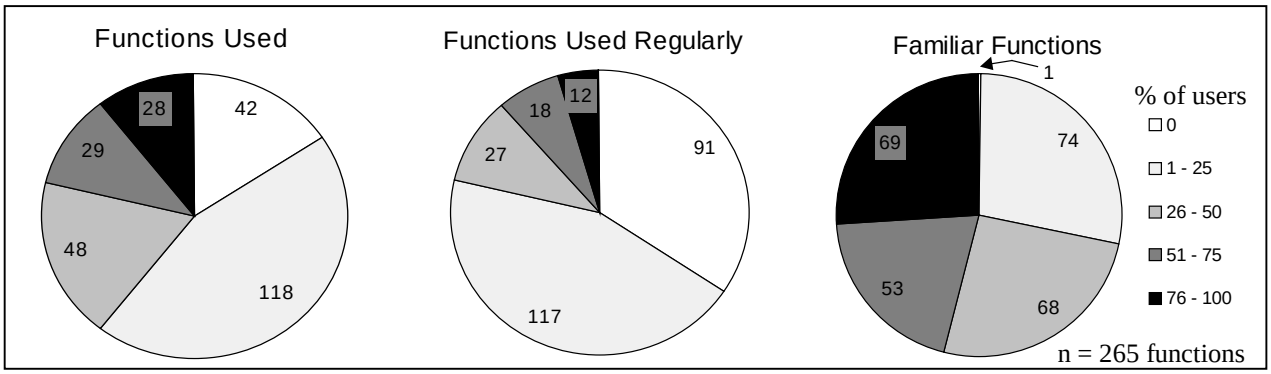

Figure 1: Number of functions that are "used", "used regularly", and "familiar" to our participants.

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. 28% (74) of the functions were familiar to 13 or fewer participants (1-25%), and 26% (69) of the functions were familiar to 40-53 (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.

We can also look at the relationship between

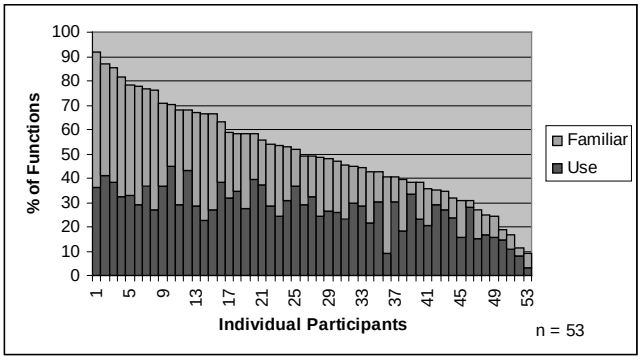

Figure 2: Percentage of functions "familiar" and "used" for each participant sorted in descending order of familiarity.

familiarity and *use* from the perspective of the individual user. Figure 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% (standard deviation = 0.148).

Table 1 shows that a relatively low percentage of the functions were actually used. On average the participants used 27% of the functions, and were familiar with 51%[4]. 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 2 shows this graphically. However, this data cannot tell us whether each user experienced the unused functionality in the same way, or whether they experienced it as "bloat".

	First-level functions	
Average # familiar to participants	135	(51%)
Average # used by participants	72	(27%)
Average # used regularly	40	(15%)
Average # used irregularly	32	(12%)
Maximum # familiar to any participant	245	(92%)
Minimum # familiar to any participant	24	(9%)
Maximum # used by any participant	119	(45%)
Minimum # used by any participant	8	(3%)

Table 1: Means and ranges of "familiar" and "used" functions (n=53).

5 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

[4] 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%.

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 categorised?
4. Did users actually perceive that 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 turn to 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 is able to deepen our understanding of users' experience and suggest design options that are masked by other methods.

5.1 Questionnaire

Participants were asked to rank on a Likert-type scale a series of 29 statements about their perception of using MSWord. Figure 3 shows the result for one of these questions, and is representative of the findings for statements on general satisfaction/dissatisfaction. Namely, 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.

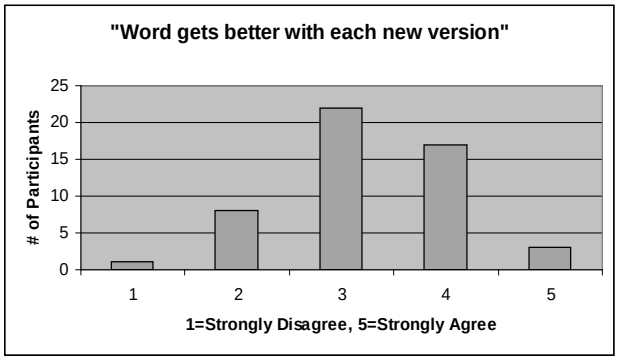

Figure 3: Participants satisfaction with MSWord.

The responses to questions designed to assess the impact of a large number of functions on usability did, however, suggest that a problem exists. Examples of these are in Table 2 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 Op.	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 2: 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). 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 Op.	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: Users' preference for number of functions on the interface.

But why are some users not bothered by excess functionality while others are? This is, in part, we argue 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.

5.2 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 summarising 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 issue.

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."*

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 three 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."*

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[5] 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."*

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

5.3 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 small 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 varies widely. This discovery of 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 in Section 7 of this paper.

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

6 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. In a workshop [5] Microsoft reported an unpublished study in which 12 members of a focus group were asked to define "bloat". Group members were asked to complete the statement "*My software feels bloated when…"* The sample is small and we have been unable to find information on how the group was chosen, but the results are intriguing. The Microsoft study reported 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. A user's profile was independent of expertise as it is traditionally defined in the HCI literature, a uni-dimensional construct which includes categories such as novice/beginner, intermediate, and expert. Microsoft's focus group included only intermediate and expert users.

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 (3.8%)! On all the questions relating to blame we had a high reporting of "no opinion", and at least two 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 4 along with the Cronbach's alpha (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 MS Word software to be up to date – I want the latest version." (# Variables = 3, α = .77)
Completeness	"I want a complete version of MS Word even if I don't use all the functions." (# Variables s = 6, α = .76)

Table 4: Summary statement for three scale variables (n = 50).

These three scales were then aggregated to comprise the final A/B Profile Scale (Cronbach's α of 0.81).

The distribution of the cases across the A/B 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 groups, which we distinguished as Profile A, Neutral, and Profile B.

We found support for Microsoft's finding that the perception of "bloat" is independent of expertise (Figure 4). In addition, we found that it is independent both in terms of the number of functions used and in terms of the number of functions with which the user was familiar.

While there is some support for the A/B profiling, 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

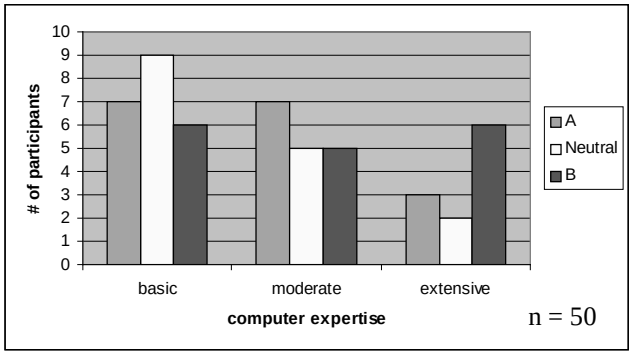

Figure 4: Distribution of computer expertise over the A/B Profile Scale.

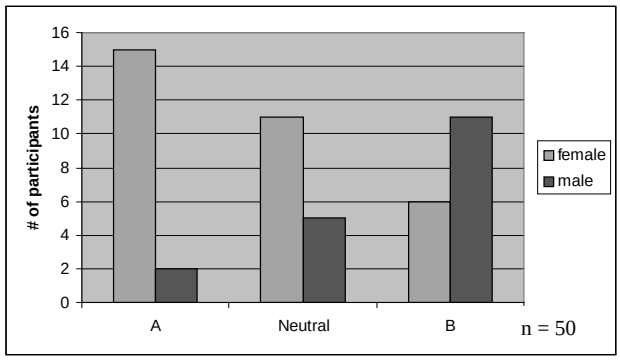

Figure 5: Distribution of gender on A/B Profile Scale.

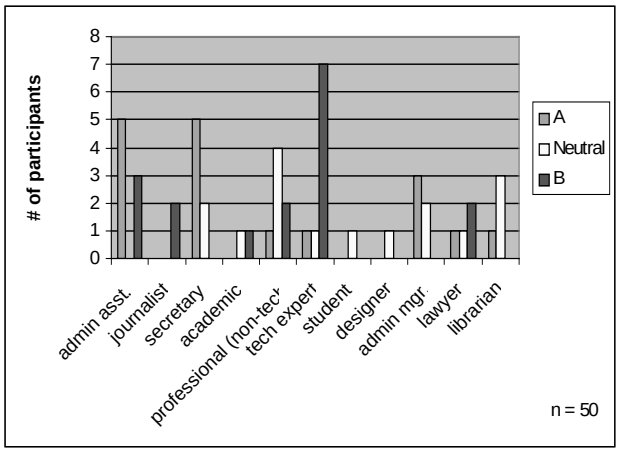

Figure 6: Distribution of occupation on A/B Profile Scale.

explaining the outcome. Figure 5 shows that there is indeed a difference based on gender, but contrary to our expectations it was the females who fell into Profile A, i.e., those wanting the most up-to-date, and complete version of the software. However, if we look at the distribution of the occupation of our participants (Figure 6), an alternate explanation is suggested. The gender difference also reflects differences in terms of their position in the division of labor.

The females cluster in administrative assistant and secretarial occupations, those for whom word processing may be the base of their craft knowledge and a source of status. Profile B includes 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 another task. Also included in Profile B are 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. The academics and technical experts are most overburdened, responsible for both the creation and production of documents. Not surprisingly, perhaps they are the least interested in new versions of the software.

7 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 is the metric, MSWord is clearly "bloated"—a little over 50% of the functionality with which users are familiar is actually used. And, while users are dissatisfied with a number of aspects of MSWord, not all this dissatisfaction centers 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 few 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.

Accepted for publication in The Proceedings of Graphics Interface 2000, Montreal, May 2000.

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 user 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.

We are starting with a two-interface model for users who want a reduced function set. This group includes beginners as well as those users who regularly use only a few features, for example, the lawyers in our study. A simple toggle that enables the user to switch between the default interface and a reduced interface – for example the top 10% of functions used by all users – is the first approach. This gives a user access to a less complex interface while at the same time permitting the user to move readily to the more complex default version, if for example, a less frequently-used feature is required. As well, features can be added into the user's personal interface as desired. In this way the complexity is masked without limiting the user's access to the full system.

Simultaneously we are investigating an alternative approach—the creation of a set of interfaces. We are exploring a number of different bases for personalization to define these sets, e.g., psychological stereotypes, social roles, activities, and digital personae. Essentially we are arguing that not only has the time for "one-size-fits-all" interface design passed, but that "one-size-fits-one", or an interface for every user, even if possible, could also be limiting. It is only with a set of interfaces that we can begin to support the complexity and diversity of users' experience. The challenge is an exciting one as applications such as the word processor are used by millions and therefore the potential impact of design changes is enormous.

8 Acknowledgements

We would like to thank CITO: Communication and Information Technology Ontario and IBM Centre for Advanced Studies who are supporting the *Learning Complex Software* Project of which this study is a part. We also thank Kellogg Booth and Ronald Baecker for commenting on earlier drafts of this paper.

References

[1] The bloatware debate (1998). *Computer World*, August 10, 1998.

[2] Carroll, J., and Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors*, 26(4), 377-389.

[3] Do computers have to be hard to use? Complex, volatile, frustrating; There must be a simpler way (1998). *New York Times*, May 28, 1998.

[4] Gibbs, W. (1997). Taking computers to task. *Scientific American*, July 1997, 82-89.

[5] Greenberg, S. (1993). *The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-based Tools*. New York: Cambridge University Press.

[6] Hanson, S.J., Kraut, R.E., and Farber, J.M (1984). Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems*, 2(1), 42-57.

[7] Kaufman, L. and Weed, B. (1998). User interfaces for computers – Too much of a good thing? Identifying and resolving bloat in the user interface. *Conference Summary, CHI 98*, Workshop #10, 207-208.

[8] Linton, F., Joy, D. and Schaefer, P. (1999). Building user and expert models by long term observation of application usage. *User Modeling: Proceedings of the Seventh International Conference*. New York: Springer, 129-138.

[9] More *than Screen Deep: Toward Every-Citizen Interfaces to the Nation's Information Infrastructure.* (1997). Washington, DC, National Academy Press.

[10] Norman, Don. (1998). *The Invisible Computer*. Cambridge, MA: MIT Press, 80.

[11] *Online Computing Dictionary* http://www.instantweb.com/

[12] Resnick, P. and Varian, H.R. (1997). Recommender systems, *Communications of the ACM*, 40(3), 56-58.

[13] Whiteside, J. et al. (1982). How do people really use text editors? *Proceedings of the Conference on Office Automation Systems*, New York: ACM, 29-40.