Accepted for publication in the Proceedings of CHI 2000, Extended Abstracts, April 2000.

"Bloat": The Objective and Subject Dimensions

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

ABSTRACT
"Bloat", a term that has existed in the technical community for many years, is increasingly receiving attention in the popular press. However, it is seldom clear exactly what "bloat" is. Our extensive study of 53 users of a complex software application, Microsoft Word, Office97, provided an opportunity to explore the concept of "bloat" in detail. We specify the concept of "bloat" and argue that it has both objective and subjective dimensions.

Keywords
Complex software, bloatware, creeping featurism, functions, office applications, user study, evaluation.

INTRODUCTION

The sharp increase in raw compute power since the PC was introduced nearly two decades ago has translated into applications with sophisticated graphical user interfaces and increased functionality. Yet we know little about how the millions of users of office applications, such as word processors, experience this increased functionality.

Recently, there has been interest in the popular press and the computer world in what has been termed "bloat" or "bloatware" and "creeping featurism" [3]. The term "bloat" has 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" [4]. Creeping featurism is the tendency to complicate a system by adding features in an ad-hoc, non-systematic manner [4]. 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 term suggesting that the software is filled with unnecessary functionality; it always has a negative connotation (e.g., [1]) but is seldom precisely defined.

What is missing is an understanding of how users actually experience complex software. Our extensive study of 53 users of a complex software application, Microsoft Word, Office 97 uncovers the users' experience. In this poster we look specifically at the issue of "bloat".

STUDY OF A WORD PROCESSOR

Methodology

The sample consists of 53 participants from the general population. All participants are users of MS Word (Office 97) on a PC. As we did not have a simple random sample of the population of MS Word users, care was taken to include subjects from a variety of occupations and organizations and from across the organizational hierarchy. Variation in terms of education and experience using computers generally and MS Word in particular was also attended to. There are two parts to the study.

Part I: Functionality Identification and Usage

The objective is to establish empirically 1) the distribution of users in terms of their familiarity with the functions in MS Word and 2) the use of functions and the variation in use across users.[1]

Functions are defined from the user's perspective rather than that of the underlying application code. Functions are action possibilities (i.e., affordances) that are specified visually to the user. The first-level count includes all icons and final menu items in the default MS Word interface. There are 265 functions[2].

In an interview subjects are presented with a series of screen shots which include all the functions. These are reviewed systematically and subjects are asked to report 1) if they know what the function does and 2) if they use it. The responses to Question 1 are *unfamiliar* or *familiar*. If subjects are familiar with a function they are asked to score usage on a 3-point scale: *regularly*, *irregularly* or *never*.

As subjects are self-reporting their familiarity with the functions, the interviewer periodically asks them to explain how a particular function works (about 1 in 10 times). This gives a measure of reliability.

Lastly, a semi-structured interview is conducted with each subject to ground, validate, and enrich our quantitative work with qualitative investigation.

Part II: Perception of Bloat

A semi-structured questionnaire is used to establish the user's level of expertise, the nature of their work practices, the type of tasks they carry out on the word processor, and their history of word processing. Finally, they are asked to evaluate a series of statements on MS Word, several of which are used to create self-reported measures of efficiency and effectiveness.

Results: Did We Find "Bloat"?

If we consider software to be "bloated" when there are a lot of unused functions, then our results show that MSWord is indeed "bloated". Figure 1 shows, for example, that there are 42 functions that were not used by any of our participants and only 12 functions that are used regularly by more than 75% of the participants. The functions that are used by very few users we designate as objective bloat.

---

[1] Seeking both the user's familiarity and use as well as studying a GUI-based application rather than a command-line system distinguishes this work from earlier investigations of function use (e.g., [2]).

[2] Detailed heuristics have been developed to count the functions. These are available from the authors. Second-level counts (first-level dialog boxes) add an additional 709 functions.

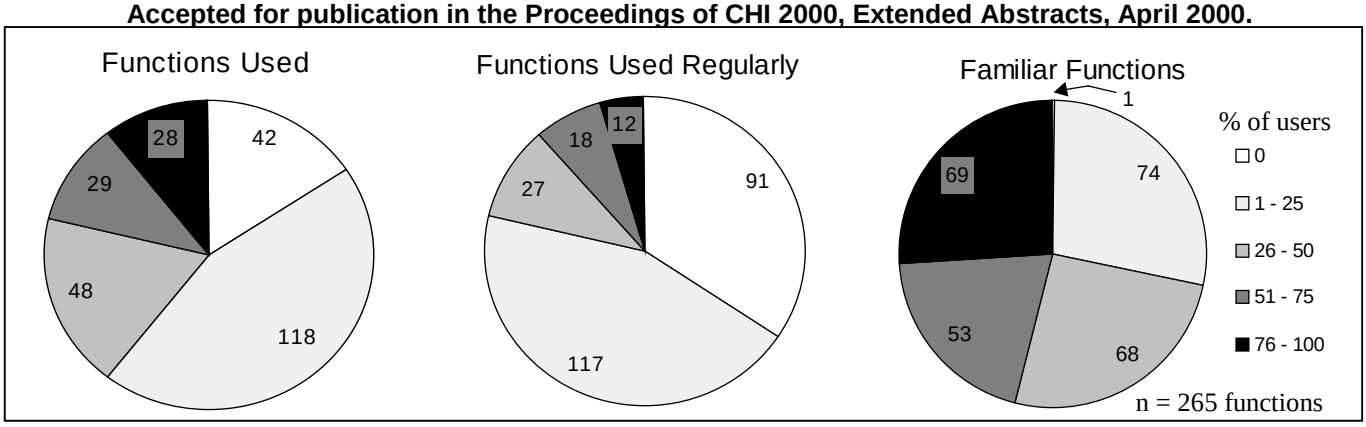

Figure 1: Number of functions that are used, used only regularly, and are familiar to our participants.

By looking at the number of functions with which users are *familiar*, however, we see that they are familiar with a great deal more than they actually use.

We can also look at the relationship between familiarity and use from the perspective of the individual user. Table 1 shows that a relatively low percentage of the functions are actually used; on average participants are familiar with 51%, and use 27% of the functions.

	First-level functions	
Average # of functions familiar to participants	135	(51%)
Average # of functions 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).

These results show that there is much unused functionality,but how do users actually *experience* this unused functionality? Our qualitative analysis provides some insight.

An especially salient finding is that not a single person in our study ever used the word "bloat", either in written comments on the questionnaire or in the interviews. The response to our milder form of the question, which inquired whether users were overwhelmed by the number of interface elements, was almost evenly divided between those who agreed, disagreed and had no opinion (Table 2). However if we ask more specific questions about function access, the distribution becomes bimodal; only 13 (25%) want to have unused functions removed entirely but 24 (45%) would prefer to have unused functions tucked away.

	Agree	No Op.	Disagree
I am overwhelmed by how much "stuff" there is.	14	20	17
I want only the functions I use.	13	5	35
I prefer to have unused functions tucked away.	24	8	21

Table 2: Perception of number of functions on the interface (n=53).

What is intriguing is that participants' responses to these statements are independent of the number of functions used, the number they are familiar with, and their level of expertise. Given this diversity, it is not surprising that there was no single group of the functions with which all participants were dissatisfied. Thus the participants differ in which functions are unused and they differ on their desire to have unused functions removed/tucked-away. This brings us to a subjective definition of "bloat" which we define as a set of functions that are not wanted and which varies from user to user.

DIRECTIONS FOR DESIGN

A design solution to the problem of "bloat" is not straightforward. The press reports that suggest that users are dissatisfied with applications such as word processors and would be better served by simple or light versions oversimplify the problem and are not grounded in actual user experience. With respect to "objective bloat" we have two basic recommendations: eliminate unused functions and relocate functions used by few users from high-level visibility in the interface. More interesting, however, and more complex in terms of design implication is what we are calling "subjective bloat". We are currently investigating ways of *masking complexity*, and looking specifically at some form of *personalisation* that is lightweight and low in overhead without adding additional complexity. We are exploring several different bases for personalisation such as digital personae, social roles, activities, etc. In terms of the interface itself we suggest that a suite of interfaces is necessary.

ACKNOWLEDGEMENTS

Gale Moore is the project leader of the Learning Complex Software project of which this study is part. We gratefully acknowledge funding from CITO, IBM CAS, and NSERC. Ron Baecker and Kellogg Booth provide graduate supervision in the doctoral program.

REFERENCES

1. The bloatware debate (1998). Computer World, Aug 10, 1998.
2. 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.
3. 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.
4. Online Computing Dictionary http://www.instantweb.com/