Published in ACM's Conference on Universal Usability, 2000, Washington, D.C., Nov 16-17.

Reducing the Gap Between What Users Know and What They Need to Know

Ron Baecker*‡, Kellogg Booth†, Sasha Jovicic‡, Joanna McGrenere‡, Gale Moore

Knowledge Media Design Institute
University of Toronto
10 Kings College Road #4306
Toronto, ON, M5S 3G4, CANADA
+1 416 978 6983
rmb@dgp.toronto.edu

ABSTRACT

"Universal usability" [17] is currently impeded by system complexity and poorly-crafted interfaces which lead to confusion, frustration, and failure. One of the key challenges is "the gap between what users know and what they need to know" [17, p. 86]. This paper describes and presents early results from three related research projects designed to identify and close this gap and to examine how users might learn what they need to know.

Keywords

Usability, human-centered design, bloat, user study, survey, email, mail system, visualization, multimedia.

INTRODUCTION

We describe three inter-related projects that address three aspects of software complexity as experienced by users: functional complexity, data complexity, and the complexity of learning about software.

The first project focuses on system complexity as manifested in functionality-filled software. This work comes out of research begun in the Learning Complex Software Project initiated in 1998 by Moore. In the first study Moore and McGrenere undertook to better understand the experience of 53 users of Microsoft Word. McGrenere, building on insights gained in this study, is designing and testing a novel system architecture to support using and learning complex software. The approach allows users to access software that concurrently embodies both what users *know* and what they might *need to know*.

The second project is an effort to enhance the usability of email systems by reducing both what users *need to know* and what they *need to do*. *TimeStor**e*** does not require users to file email in folders. Instead it presents a novel interface for managing large bodies of email and retrieving needed messages through a triad of automatically derived email descriptors: *when* received, by *whom* sent, and dealing with *what*. We summarize recent results of Jovicic, who built and tested a new Java implementation of TimeStore.

The third project focuses on the development of a new method for mitigating system complexity by radically changing the way in which we present information about what users need to know. The method borrows from the field of software visualization. It uses dynamic graphic presentations to display software behaviour, but focuses on showing *users* what to do, rather than showing *programmers* how programs operate, as in conventional software visualization. We replace typical text and still-graphic forms of documentation and on-line help with structured video explanations that live on the Web and can be streamed over the Internet "just in time."

All three projects deal with the management of some form of complexity. The first project addresses the complexity of having a plethora of functions in the interface. The second project addresses the complexity of managing large quantities of email data. The third project proposes the use of video explanations as a means of mastering complexity by presenting users information that they need to know in a comprehensible form when they need to know it.

The rest of this paper is organized as follows. Each of the projects is described in turn, starting with a presentation of the appropriate research context. The results of the projects are then integrated by extracting common research themes, describing the next steps in the research, and identifying common research challenges that must be addressed if we are to achieve the goal of universal usability.

LEAVE BLANK THE LAST 2.5 cm (1") OF THE LEFT COLUMN ON THE FIRST PAGE FOR THE COPYRIGHT NOTICE.

* Also with Expresto Software Corporation, Toronto

† The Department of Computer Science, University of British Columbia

‡ Also with the Department of Computer Science, University of Toronto

PROJECT 1: COPING WITH FUNCTIONAL COMPLEXITY

End-user applications have changed dramatically since the PC and the Macintosh were introduced two decades ago. A sharp increase in compute power and strong market forces have resulted in desktop applications, such as word processors, with sophisticated graphical user interfaces and with considerably more functionality than their predecessors. The assumption has been that the more functionality there is, the more useful and marketable the application will be. As these applications have become more powerful and more complex, there has also been a substantial increase in both the number and the diversity of users. Users differ in their knowledge of the various skills needed to use an application and the tasks that they do, and hence in the knowledge they need. Yet desktop applications typically present a single user interface that is expected to accommodate all users. If universal usability is to be achieved with general productivity applications, we need to challenge the notion of the "one-size-fits-all" interface.

The term "bloatware" or "bloat" has been used within the technical community [11] and in the popular press [7] to describe heavily-featured applications. "Bloat" is seldom clearly defined, but used as a catch-all phrase indicating that an application is filled with "unnecessary" functionality. Do all users experience such software as bloated? If not, how do users actually *experience* heavily-featured software? These questions have not been addressed in the research literature, yet could provide valuable insights for interface design. Our extensive study of 53 users of a complex software application, Microsoft Word, Office 97 (MS Word) helps us to understand how users experience this complex software application. The study is outlined below and selected results are highlighted, looking specifically at the gap between what users know, what they need to know, and what they think they need to know. The reader is encouraged to read McGrenere and Moore's paper [15] for a more complete discussion.

Study of a Word Processor

In order to capture the diversity of user experience it was important to identify and select a group of people who were representative of the general population of MS Word users and not simply a sample of convenience. Care was taken to achieve representation in terms of variables such as age, gender, education, occupation and organizational status. Participants also varied in their experience with computers and with MS Word. The final sample consisted of 53 participants. Two parts of the study are relevant here.

Part I: Functionality Identification and Usage

The objective of Part I was 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.

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 included all icons and final menu items in the default MS Word interface. We counted 265 functions[1].

In interviews subjects were presented with a series of screen shots which included all 265 functions. These were reviewed systematically and subjects were asked to report (1) if they were familiar with what the function does and (2) if they used it. The responses to (1) were *unfamiliar* or *familiar*. If subjects were familiar with a function they were asked in (2) to score usage on a 3-point scale: *regularly*, *irregularly*, or *never*. Note that we specifically chose self-reporting over software logging which has generally been the method of choice in computer science for capturing function usage [e.g., 9, 13]. The reason is that logging cannot distinguish between *familiarity* and *use* and must be used for an extensive period of time if irregularly used functions are not to be missed.

Part I concluded with a semi-structured interview with each subject to ground, support, and enrich our understanding of the users' experience. The result is that the quantitative findings can be contextualised by qualitative interpretation.

Part II: Perception of Bloat

A semi-structured questionnaire was used to establish the users' levels 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 were asked to evaluate a series of statements about MS Word, several of which were used to create self-reported measures of efficiency and effectiveness.

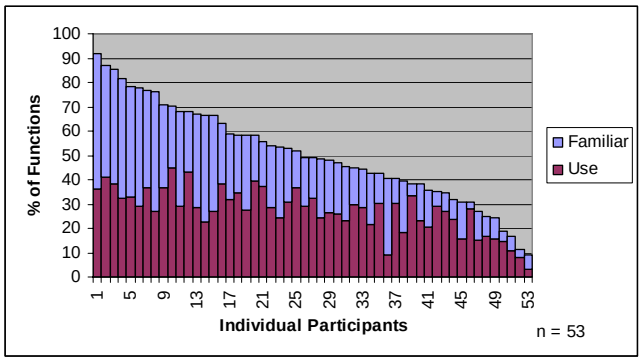

Figure 1: Percentage of functions "familiar" and "used" for each participant sorted in descending order of familiarity.

Selected Results

If software is considered to be "bloated" when there is a large number of unused functions, then our data shows that MS Word is indeed "bloated". For example, of the 265 first-level functions, there were 42 that were not used by any participant, 118 that were used by less than 25% of the participants, and only 12 functions that were used regularly by more than 75% of the participants. We can also look at

[1] 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.

the usage and familiarity data from the perspective of the individual user (Figure 1). On average the participants used 27% of the functions, and were familiar with 51%. The mean Usage to Familiarity Ratio was 57% (standard deviation = 0.148). One way to interpret this result is that users know roughly twice as much as they need to know, at least if we measure need-to-know by the functions they actually use in practice. Those functions that are not used by any users provide an objective measure of "bloat".

But do users actually experience the software as "bloated"? Do those who use very little functionality experience the excess unused functionality negatively? Do those who use most of the functionality experience MS Word more positively? Our qualitative data from Part II provides some insight on this issue.

Responses to questions designed to assess the impact of a large number of functions on usability do indicate that some users have a negative experience (Table 1). 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 polarized.

	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 1: 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 2). 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 2: Users' preferences for numbers of functions in the interface.

Contrary to our prior assumptions, participants' responses to these statements (Table 1 and Table 2) are actually independent of the number of functions used, the number they are familiar with, and their level of computer expertise[2]. Given this diversity, it is not surprising that there was no single group of functions with which all participants were dissatisfied. This led us to define a subjective dimension of "bloat": users differ in the functions that they know and use and they differ in their desire to know about unused functions and have them available in the interface. The fact that some users do not experience unused functionality negatively should be respected in future interface designs. Further in this paper we describe one such design that we have prototyped and are evaluating.

PROJECT 2: COPING WITH DATA COMPLEXITY

The second study deals with the complexity arising from overload of email in the modern workplace. The increase in volume of email makes it apparent that semantic hierarchies of files and folders, the currently predominant paradigm, are not suitable for organizing all electronic information. Numerous studies document users' frustration and inability to organize their email effectively [4, 12, 22]. The complexity that arises in attempting to manage the high volume of email needs to be more adequately addressed.

The Design Approach

The TimeStore project [14, 18, 23, 24, 10] addresses the complexity involved in managing large volumes of email. Here semantic hierarchies are abandoned in favour of an organizational approach which eliminates the need for filing; messages are automatically organized by time and by sender in a two-dimensional grid.

Jovicic [10] reviews relevant literature on human memory in order to determine how the email retrieval process might be supported. She makes recommendations for the design of user interfaces for managing large amounts of email based on what the user knows (remembers) about the context in which the email arose. The advantage of this approach is that the user need no longer construct and maintain a semantic hierarchy for organizing and managing her email, nor need she know (remember) where in the hierarchy she stored a particular message.

Background

This context of a piece of email can differ widely from message to message. Some messages may best be regarded as *autobiographical* (personal) *events,* and some as *news events* (events where a person was not present). Some may have elements of both types of events, in that they deal with news that has a direct personal connection to the recipient and to events in his or her life.

[2] The two exceptions to this are that users who want only those functions that they use tend to be those with more computer expertise, and the users who have a hard time finding the functions tend to be those users who are familiar with and use relatively fewer functions.

Memory literature indicates that the context of recent autobiographical events — the people present, the location of the event, and the time of the event — are typically well-remembered [20]. With old autobiographical events, contextual information is gradually lost over time. With news events, little contextual information is available.

Other memory literature results (such as [8]) concern temporal schemata such as days, weeks and years, which are commonly used in event retrieval. The importance of weeks is especially salient, since day-of-week schema are strikingly resilient to the passage of time. That is, people can locate the exact day of the week for an event, even if they cannot locate the week of the event.

Design Details and Implementation

In order to reduce the complexity of handling email messages, TimeStore does not require the users to do any filing. Instead, email messages are displayed in a two-dimensional grid organized by time and sender (Figure 2). Memory literature stresses that information about time and place of the event, the people involved, and the main activity are all well remembered. All of these components except place have a straightforward analogue in email messages, and are incorporated into the user interface. The two-dimensional representation allows locating messages by specifying *when* the message was received and by *whom* it was sent. In order to make it possible to locate messages by content (*what* it's about) as well, TimeStore allows narrowing of the search space using full-text searching by one or more keywords.

Figure 2: The user interface of the TimeStore prototype. A dot shows that messages were received from that sender on that day. Clicking on a dot pops up a window with a list of messages.

The search engine also compensates for lack of context, which is particularly important for old autobiographical events and news events, both of which have little contextual information.

Weekends are emphasized by dark bands that visually separate weeks. As noted earlier, days of weeks of events are well remembered and often used in dating, so their use is supported in the interface. The senders, represented along the vertical axis, may be clustered in groups. This provides additional retrieval context and makes it easier to locate messages by sender.

The interface represented in Figure 2 was implemented using Java and C and integrated with the Eudora email system. Preliminary user testing [10] shows that users liked the visualization of their email and found it useful for retrieval of both old and new messages.

PROJECT 3: LEARNING ABOUT COMPLEXITY

The third project takes a different approach to complexity and the gap between what users know and need to know by focusing on the method by which features are explained.

Various methods are used today without great success to introduce and explain complex technology such as software. Documentation typically consists of lengthy prose interspersed with screen snapshots, but users don't read manuals and typically find "online help" unhelpful. Vendor support staff and corporate help desks struggle with creating and conveying answers to queries, but it is difficult to explain complex step-by-step procedures in words. Courses present lots of material, but usually not what is needed when it is needed. Videos are hard to search and often out-of-date. One-on-one tutoring is best because it allows software to be demonstrated, but is expensive and rarely available when needed.

We describe here an alternative approach which draws from the field of software visualization [19]. *Software visualization* has been defined [16] as "the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software." To date, almost all work in the field has focused on enhancing the human understanding of how software operates internally (for example, [1, 2, and 5]).

Our approach is to facilitate the rapid development of structured digital video presentations that demonstrate software and show users how to accomplish desired tasks. This approach is supported by a recent comprehensive review of the instructional effectiveness of video media [21], which suggests that video's strengths arise in situations where "[it] might provide additional visual forms of information to that available in descriptions given in text..., and when learning procedural sequences might be benefited by conveying motion video compared to static or verbal descriptions" [21, p. 210].

The video demonstrations and explanations are then integrated into the training, support, and sales sections of a company's Web site. They can be accessed from the Web by users and streamed over the Internet "just in time." Current technology easily supports this and the ubiquity of Internet access makes it a practical solution for many users.

The Authoring Technology

The video authoring system, Expresto Creator/Publisher, enables the rapid creation and desktop publishing of video communications. The system, previously called the

University of Toronto Movie Authoring and Design system in published papers such as [3], is a thinking tool, a "word processor for movies," which allows great ease in imagining, structuring, creating, and improving software explanations. It enables automatic publication of media on Web sites, and transmission over the Internet as streaming video and as video email. The Web publishing ability allows integration of video communications with existing corporate databases and Web applications such as customer care and online learning solutions.

Expresto's system makes this new software explanation medium possible, and makes it easy, rapid, and cost-effective. It does this by:

- supporting the entire media creation process
- encouraging a structured design process to allow the hierarchic organization of a video communication, and nonlinear access to its component parts
- supporting the integration (via the Web) of other explanatory information with each segment of these structured, randomly accessible, video communications.

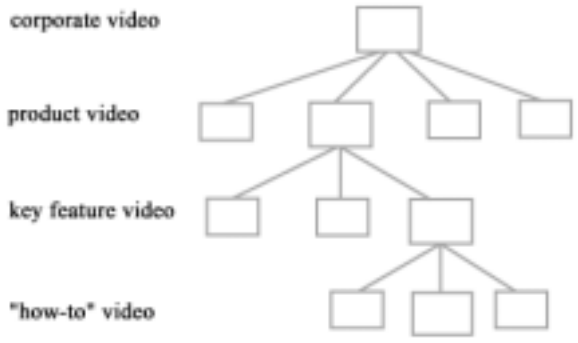

Figure 3: Hierarchical structure of videos explaining CaseWare's products.

A Typical Project Example

To illustrate the approach, we briefly present our work with CaseWare International, a Toronto-based firm that develops software for auditors and markets it worldwide. Under development is a series of short videos based on the hierarchical structure that appears in Figure 3.

The top-level of this structure is a corporate video describing the capabilities and strengths of CaseWare. The next level comprises a set of product videos introducing the company's products. The third level focuses on key features of the products, and the fourth level presents explanations of how to accomplish tasks that exploit the key features. The top three levels are useful in software sales, and the bottom three in support and training.

Figure 4 shows a typical display presented to a user viewing a movie about the "mapping" key feature of CaseWare's "Working Papers" software. The movie appears on the left, and a list of "show me how to" movies appears on the right. A frame from one of these "show me how to" movies appears in Figure 5.

RESEARCH THEMES AND CHALLENGES

These projects, while diverse, all illuminate different facets of the problem of software complexity: functional complexity, data complexity, and mastering complexity. Additional links among the projects appear as common research themes, next research steps, and challenges.

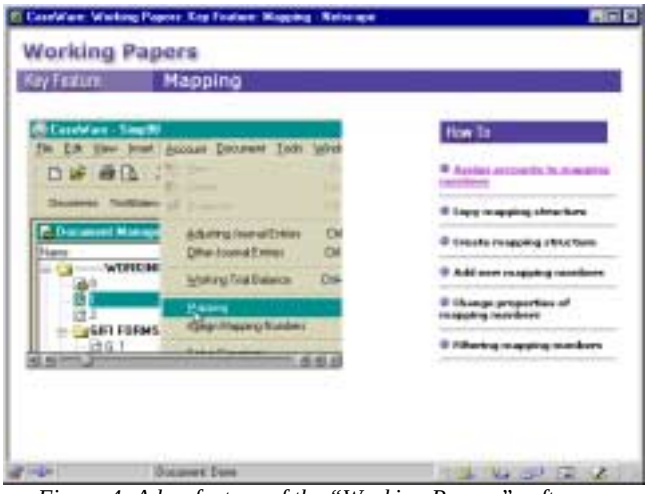

Figure 4: A key feature of the "Working Papers" software.

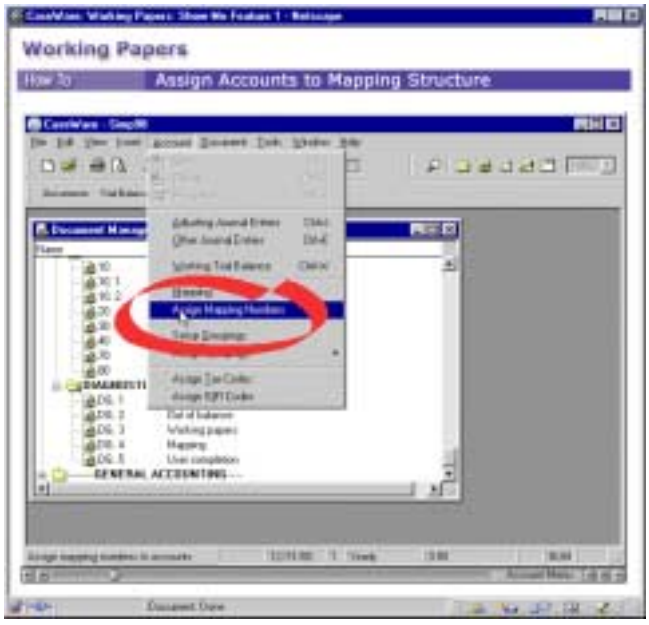

Figure 5: A movie explaining how to accomplish a task with a key feature of the "Working Papers" software.

Themes

All three projects deal with the management of complexity. Doing so requires the development of appropriate and in some cases novel *representations*, such as the use of video in place of text to explain how to use software, or the display of large bodies of email as two-dimensional tableau rather than as conventional lists of messages.

Innovations in representing complexity also encourage us to provide *multiple views* and *multiple control mechanisms* for transacting with complex systems. Thus we interact with

email both through Eudora lists and the TimeStore tableau. We also envision a word processor in which people can work concurrently with a conventional feature-rich version and a stripped-down version that includes only those functions that they currently know and use.

Directions for the Next Steps in the Research

Interface designers have begun to recognize that "one-size-fits-all" interfaces may not in fact fit all and have provided facilities for customization and tailoring. Our first project showed that such facilities are too heavyweight and are thus ineffective because users are constantly confronted with the gap between what they know and what they need to know. We recommend that interface design should support the creation of a personalizable interface in a way that is low in overhead for the user. We suggest that one way to start is to provide multiple interfaces where individual interfaces within the set would be designed to *mask complexity* and support learning through a lightweight mechanism for moving among these interfaces.

To test this hypothesis, we have built a simple three-interface MS Word prototype that provides one interface with a minimal function set (similar to MS NotePad), a second interface with a function set personalized to the user's needs and wants, and a third interface with the default function set. We have tentatively called these interfaces "Mini Word", "My Word", and "Maxi Word". Users can easily toggle between the three interfaces and in this way are not limited to a pre-selected set of features. In the current prototype it is the experimenter who constructs and adapts the "My Word" version on behalf of the users through a Wizard of Oz methodology. This particular design is a direct response to the understanding gained in the initial study, which suggested that users want both simplicity ("what they know") and access to additional functionality ("what they may need to know"). Our prototype accommodates both novice users and users who regularly use only a few features, for example, the lawyers in our study. This general direction for our design is similar to much earlier work by Carroll and Carrithers, whose "Training Wheels" interface [6], despite its promise, has never been fully actualized.

Our multiple-interface approach, a response to subjective bloat, is a generalization of Carroll and Carrithers's notion that learning should be integrated with use so that users can actively manage the gap between what they know and what they need to know. Switching between multiple interfaces does not, however, tell us how learning can best be achieved in this context.

The third project tackles that problem, taking advantage of digital video technology, something not available when "Training Wheels" were first investigated. This approach may appear to be a radical one, especially given the current costs associated with digital video, but these costs must be compared with current organisational training and support budgets, which are enormous.

Video explanations may be more expensive to produce than text, and certainly they consume more resources to store and deliver. The latter two will diminish as Internet access drops in cost and increases in bandwidth. Future work will seek to generate experimental evidence of the relative communication effectiveness of video versus text plus still graphics for demonstrating and explaining software. Other near-term goals are more pragmatic, and center around improvements in production efficiency and the development of methods for managing, accessing, and serving large bodies of video material.

The second project in fact addresses the latter issue, but in a different context. TimeStore's novel view facilitates access to large bodies of email. The approach would become even more interesting with the incorporation of "where" as well as "when", "who" and "what" to provide a fuller context for email retrieval. Many people read their email from multiple locations. If the software were aware of location (which it easily could be), this would provide retrieval cues related to "where" a message was read — a component of autobiographical memory that is currently not utilized. This will be powerful and essential as the use of various mobile devices such as PDAs and Web-enabled cell phones becomes more widespread.

Integrating ideas from the three projects leads us to consider future systems that will have multiple interfaces that will permit a user to easily navigate between what the user currently knows and what the user needs to know. Our current MS Word prototype does this with three interfaces, but leaves open the questions of how the user will learn about unknown features and how to facilitate the construction of the personalized interface, namely "My Word". The third project provides a mechanism to address the learning question through just-in-time digital video explanations. These explanations then become part of the complexity problem themselves, since the user needs to manage knowledge of which video explanations they might need to return to for additional help. The second project provides a promising solution by incorporating full "what", "when" and "where" contexts, which could help in managing multiple interfaces and multiple help systems.

Challenges for Universal Usability

Developing experimental software to manage complex sets of functions or data, as we have in these three projects, requires us to construct *scalable* experimental systems capable of handling this complexity. We therefore face software engineering challenges not always present in experimental computer science, in which new ideas can be explored on toy problems and data sets. For example, in the first project we had to instrument MS Word to toggle between a user's personalized function set and the full functionality of Word. We were able to accomplish this using Visual Basic scripting, but this placed limitations on what we could do. In general, we believe that systems should be designed to support multiple interfaces. Developing a general architecture that encourages this design discipline is the on-going focus of the first project.

These experimental systems must also be tested *in real use by real users in real work contexts over extended time frames*, and not in artificial laboratory environments. The relationship of technology to *work practice* is paramount, and we employ a human-centred design process. Social and ethical issues often arise, such as protecting the privacy of our users when studying the impacts of new technology on how they handle their email. In our second project, we had to work with users and their actual email system in order for our study to be meaningful. We did this by building the TimeStore software on top of Eudora taking advantage of Eudora's API. In order to broaden our subject pool for future studies, we will need to consider fitting TimeStore to other commercial email products.

Projects such as these stress our *evaluation methodologies* to the limit, as they require field studies with extensive use of both quantitative and qualitative research methods. For example, our second and third projects have thus far been limited to in-house testing, but a strength of the first project is that we conducted a study in the workplace and triangulated multiple methodologies. The triangulation of methods provides several perspectives as each method helps illuminate a different part of the problem.

A significant impediment to universal usability is the complexity inherent in many of today's software systems. Complex functionality, data complexity, and the complexity of learning about these systems all affect usability. The use of multiple interfaces, customized for what the user needs to know, is intended to reduce the functional complexity seen by users. Introducing "who-what-where-when" information that the user already knows is intended to reduce data complexity. The use of just-in-time digital video explanations is intended to reduce the complexity of learning software.

ACKNOWLEDGMENTS

We gratefully acknowledge research support from the Natural Sciences and Engineering Research Council of Canada, from Communications and Information Technology Ontario, and from the IBM Canada Centre for Advanced Studies. Agnes Ouellette and Anne Dmitrovic of Expresto Software contributed to this work and helped prepare the paper.

REFERENCES

[1] Baecker, R. (1998). *Sorting Out Sorting*: A case study of software visualization for teaching computer science. In [19], 369-381.

[2] Baecker, R. & Marcus, A. (1990). *Human Factors and Typography for More Readable Programs*, ACM Press.

[3] Baecker, R., Rosenthal, A., Friedlander, N., Smith, E., & Cohen, A. (1996). A multimedia system for authoring motion pictures, *Proc. ACM Multimedia'96*, 31-42.

[4] Balter, O. (1998). Email in a working context. *Doctoral dissertation.* Royal Institute of Technology, Sweden.

[5] Brown M.H. (1988). *Algorithm Animation*. MIT Press.

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

[7] *Computer World*, Aug 10, 1998. The bloatware debate.

[8] Friedman, W. J. (1993). Memory of time for past events. *Psychological Bulletin*, 113(1), 44-66.

[9] Greenburg, S. (1993). *The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-base Tools.* Cambridge University Press.

[10] Jovicic, A. (2000). Implications for the design of email management software. *M.Sc. thesis*, Department of Computer Science, University of Toronto.

[11] Kaufman, L. & 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.

[12] Lantz, A. (1998). Heavy users of electronic mail. *International Journal of HCI*, 10 (4), 361-379.

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

[14] Long, B. (1994). TimeStore: Exploring time-based filing. Unpublished study, University of Toronto.

[15] McGrenere, J., & Moore, G. (2000). Are we all in the same "bloat"? *Proc. Graphics Interface 2000*, 187-196.

[16] Price, B., Baecker, R., & Small, I. (1993). A principled taxonomy of software visualization, *Journal of Visual Languages and Computing*, 4(3), 211-266.

[17] Shneiderman, B. (2000). Universal usability. *Communications of the ACM*, 43(5), 85-91.

[18] Silver, N. (1996). Time-based visualizations of electronic mail. *M.Sc. thesis*, Department of Computer Science, University of Toronto.

[19] Stasko, J., Domingue, J., Brown, M., & Price, B. (1998). (Eds.), *Software Visualization: Programming as a Multimedia Experience*. MIT Press.

[20] Thompson, C., Skowronski, J., Larsen, S. & Betz, A. (1996). *Autobiographical Memory: Remembering What and Remembering When.* Erlbaum.

[21] Wetzel, C., Radtke, P., & Stern, H. (1994). *Instructional Effectiveness of Video Media*. Erlbaum.

[22] Whittaker, S. & Sidner, C. (1996). Email overload: exploring personal information management of email. *Proceedings CHI '96*, 276-283.

[23] Yiu, K. (1997). Time-based management and visualization of personal electronic information. *M.Eng. thesis*, Department of Electrical and Computer Engineering, University of Toronto.

[24] Yiu, K., Baecker, R., Silver, N., & Long, B. (1997). A time-based interface for electronic mail and task management. *Proc. HCI International '97*, 2, 19-22.