Responding to cognitive overload: Co-adaptation between users and technology

Wendy E. Mackay, Visiting Professor

Department of Computer Science, Aarhus University
Ny Munkegade, building 540, DK-8000 Aarhus C, DENMARK
mackay@daimi.au.dk

ABSTRACT

This study examines how the first users of the X-Window system, the administrative staff at MIT's Project Athena, coped with this complex new technology. By examining their use of customization files over a period of six months, we were able to track how they responded to both organizational and technological changes. We found clear evidence of co-adaptation: individuals both adapted to the new technology, influenced by its design, implementation and use within the local work environment, and they adapted it for their own purposes, reinterpreting it in ways unanticipated by the technology's designers. This study provides evidence that users' co-adaptation of technology is a useful mechanism for addressing cognitive overload. It also suggests how taking co-adaptation into account when designing new technology can help users manage the growing complexity of their work environments and the corresponding increase in cognitive overload.

Keywords

Co-adaptive systems, customization, user innovation, participatory design, information filtering

Réagir à la surcharge cognitive : co-adaptation entre utilisateurs et technologie

Wendy E. Mackay, Visiting Professor

Department of Computer Science, Aarhus University
Ny Munkegade, building 540, DK-8000 Aarhus C, DENMARK
mackay@daimi.au.dk

RESUMÉ

Cette étude analyse comment les premiers utilisateurs du système de fenêtrage X-Window System, l'équipe administrative du projet Athena au MIT, ont abordé cette nouvelle technologie. En examinant leur utilisation des fichiers de personnalisation du système sur une période de six mois, nous avons pu suivre comment ils ont réagi à des changements à la fois organisationnels et technologiques. Nous avons mis clairement en évidence un phénomène de co-adaptation: les individus se sont adaptés à la nouvelle technologie, influencés par sa mise en oeuvre et son usage dans l'environnement local, et ils l'ont adaptée à leurs propres besoins, la réinterprétant sous des formes non anticipées par leurs concepteurs. Cette étude fournit une preuve que la co-adaptation des utilisateurs à la technologie est un mécanisme utile pour gérer la surcharge cognitive. Elle suggère également que la prise en compte de la co-adaptation lors de la conception d'une nouvelle technologie peut aider les utilisateurs à gérer la complexité croissante de leur environnement de travail et l'augmentation correspondante de la surcharge cognitive.

Mots-Clés :

Systèmes co-adaptatifs, personnalisation, innovation par les utilisateurs, conception participative, filtrage d'information

INTRODUCTION

Although computer technology is usually justified in terms of increasing human productivity, it often has the opposite effect, creating more complex work situations that reduce productivity and increase cognitive overload (Landauer, 1986). Building more effective technologies requires a better understanding this complex relationship between technology and its use. Orlikowski (1992) identifies two basic research strategies: the "organizational imperative" in which decision-makers appropriate technology, and the "technological imperative" in which the technology influences the actions of people in the organization. Zuboff (1988) provides an extensive review of the former approach. She argues that managers choose between technologies that "automate" work and those that "informate" work: the former treats users as simply another component in the work process whereas the latter empowers them to make decisions and operate autonomously. Other researchers emphasize the effect of new technology on the organization. For example, Barley (1986) showed how the introduction of identical CT scanners resulted in different organizational structures in two Radiology departments. Studies by Sproull and Kiesler (1986) and Eveland and Bikson (1988) both found that the introduction of electronic mail changed the nature of communication within the organization, affecting both the organizational structure and the actual information conveyed.

These studies analyze human behavior at the organizational level: Some groups, i.e. managers and technology support, specify the technology to be used by other groups of employees. Yet organizations are composed of individuals, each of whom must cope with changing technology in their own ways. Are these individuals simply passive recipients of new technology, with their use dictated entirely by organizational rules and the system design? A two-year study of an early electronic mail filter (Mackay, 1988) demonstrated that individuals often re-interpret and sometimes actively change technology. The Information Lens was originally presented as a sort of "automatic secretary" designed to prioritize messages prior to the user seeing them. Several individuals discovered a way to run filtering rules after reading their messages, effectively creating an automated filing system. This re-interpretation of the system caused the system developers to re-design the next version of Lens, explicitly giving users multiple rule sets to accommodate both the "automatic secretary" and the "automatic filer" usage models. Users again re-interpreted the system: They created specialized rulesets to accommodate changes in their work context. Thus "vacation rules", used after an absence of a week or more, would aggressively delete messages that would otherwise have been retained. The Information Lens study showed that individuals do not simply respond to technology, they re-interpret it and adapt it for their current needs, often in ways unanticipated by the designers of the system.

This phenomenon is called *co-adaptation* , Mackay (1990a), because individuals both adapt to the technology, but also adapt it for their own purposes. The term is influenced by the related phenomenon in evolution. Until recently, scientists examined how the environment affected plants and animals, treating it as an independent variable influencing evolution. Later researchers, such as Lovelock (1979) and Margulies (1986), pointed out that living organisms actively change the environment, as well as react to it. Thus, for example, the composition of today's atmosphere is due to a complex interaction between living organisms and the physical earth, over millions of years. Successive generations of plants and animals changed the atmosphere to the point that we, as human beings, can now exist. The term *co-evolution* captures the idea of these two interdependent cycles of change. Since human responses to technology occur within a much shorter time-frame, I have used the term co-adaptation instead.

Clearly not all new technology results in overload situations and equally clearly, some individuals are better at coping than others. This paper examines how members of the administrative staff at MIT's Project Athena successfully (or unsuccessfully) co-adapted to the constantly changing X-Window system, via use of customization files. The paper then suggests how technology designers can explicitly take co-adaptation into account when creating new technology, providing users with on-going support for adapting the technology to meet their current needs.

CUSTOMIZATION AT MIT'S PROJECT ATHENA

The X-Window System, developed at MIT's Project Athena, has become a world-wide standard for Unix workstations. This study examines how the first group of users, the administrative staff at Project Athena, coped with incessant technological change and extreme cognitive overload by customizing their software environments. Customization files provided an unusually good dataset for studying co-adaptation because individual patterns of use were encoded and continued to influence each user's behavior over time. The range of possible customizations was constrained by the software design, but could also be modified by users in unanticipated ways, as they appropriated the software for their own purposes. Because customization patterns are recorded in files that can be shared among users, customizations often served to informally establish and perpetuate group norms of behavior. Since these patterns were encoded naturally, they offered an important record of customization patterns as they changed over time.

Research Setting: MIT's Project Athena

Project Athena was an "experiment in educational computing" at the Massachusetts Institute of Technology (MIT). Sponsored jointly by Digital Equipment Corporation and IBM in 1983, the eight-year, $100 million project resulted in several world-wide software standards including the X-Window System (Scheiffler & Gettys, 1986) and influenced the

strategic direction of the computer industry (Lampe, 1988, Champine, 1987). In 1990, Project Athena was the world's largest centrally-administered distributed computer environment, with over 1000 high-performance workstations distributed throughout the MIT campus.

Research Method

The purpose of the study was to provide an in-depth look at the customization activities of active users of the Athena software environment. Project Athena's staff were of particular interest because they were lead users (von Hippel, 1986) of a work environment that is now common place. Staff members were the first to experience problems and had the power and resources to make innovative changes. It is important to emphasize that, although they are located at a University, the Athena staff's deadlines were real. A software company may "slip" a hardware or software release date, but MIT never slips the beginning of the semester. An error in a software release affected thousands of people. The MIT community is both forgiving (students graduate, encouraging early mistakes to be forgotten) and critical (members of this community were articulate in criticizing Athena on a number of levels) (Turkle, 1984). The MIT culture supports a "let many flowers bloom" philosophy which encourages diversity. There is a corresponding "survival of the fittest" philosophy, in which only the best survive. Both of these philosophies are reflected in the projects and software supported by Project Athena.

Participants

The Project Athena staff consisted of approximately 80 people during the course of the study, including managers, secretaries, technical and non-technical staff. They provided a variety of services, similar to the MIS department of a large corporation. Over 60 staff members participated in some part of the study, but several left Project Athena and several did not complete all of the questionnaires. 51 people completed all of the interviews and supplied all of the requested data. The study participants included a cross-section of managers, administrative personnel and both technical and non-technical individual contributors.

All members of the Athena staff had at least one workstation in their offices. Staff members had individual accounts, could access the internet, and had disk quotas for backed up file storage. Major upgrades were tested on the staff first and then introduced at the beginning of each semester and at the beginning of the summer term.

Customization environment

Unlike the Apple Macintosh’s unified user interface philosophy, Unix and the X-Window System provide many user interface choices. In fact, no individual has complete control over the interface: it is determined by decisions made by system programmers, system administrators, application developers and the end user. The result is a highly flexible but also sometimes unpredictable environment (Norman, 1981). In order to protect themselves,

users would minimize confusion and reduce overload by creating standard customizations across applications. Users could express preferences at different levels, including how an application looks (e.g. font sizes, borders, colors, shapes) and how to interact with it (mouse, key bindings, menus, etc.) Some choices affected all applications, such as the choice of a window manager or the use of "X resources". Others were specific to an application. Customization was generally accomplished by editing a separate file (referred to as a "dot file"). Some users were unaware of the existence of these files, which did not appear in a normal directory. These files can be edited with any text editor and are executed whenever the application is run. Users could exchange parts or all of these files with each other by copying them or via electronic mail. Some customizations were highly visible and might be noticed by someone walking by, such as an unusual pattern on the background screen. Other items were less noticeable, particularly choices of keys and specifications about process, and would only be noticed if someone was watching the user carefully and noticed a difference from the observer's own use.

Data

The data consist of open-ended interviews (conducted in several iterations over four months), questionnaires, and automatic records of customization activities, in addition to informal discussions about customization with some participants. Prior to each interview, participants were asked to fill out a two-page questionnaire with background information, e.g. their programming backgrounds and current job responsibilities, information about which software they use, which applications they customize and how much, and the sources of information they use to find out how to make a particular customization. Participants then filled out two additional questionnaires, during or after the interviews, including information about:

1. Sources of information about customization
2. Levels of use of different Athena applications
3. Levels of customization of different applications
4. Levels of conversation with other staff members
5. Sources and recipients of customization files

The latter two questionnaires were modeled after the sociogram devised by Allen (1972) for the purpose of identifying communication networks within an organization. Cross-checks were made to see whether people who were identified as borrowers identified themselves as having borrowed the files, and vice versa. Customization files often contained a header that identified the file's creator and the people who had subsequently modified it. The following records were also extracted directly from the workstation:

1. A file with the dates of all system upgrades for that workstation.
2. A list of customization files, with sizes and modification dates, ordered by date.
3. Selected customization ("dot") files.
4. The standard screen layout.

5. A list of current aliases.
6. Protection status of files (whether open or closed).

This data captured most, but not all, of the sharing within the organization. For example, one person might try a feature she noted on another person's screen and then delete it again before the next data sample was taken. Also, people did not always remember the sources and recipients of customization files, especially if the exchange occurred months or years ago. People tended to remember who spent a great deal of time helping them, but not someone that they borrowed something from on the spur of the moment. Thus, these data under-represent the level of sharing within the organization. Additional data included the staff mailing list, informal discussions with staff members, a review of the on-line consulting system logs (which included hundreds of questions about customization), organization charts and a list of office changes.

Interviews were conducted in each participant's office to help trigger their memories and to make it easier to ask questions about particular files or customization activities. Each participant was asked to print out a second copy of the ordered list of customization files, which provided an indication of the rate of customizing and identified which files had been changed. Participants were then asked to show their customization files, describe the reasons for the customizations and explain the circumstances under which they were made, particularly if they were borrowed from or given to another person. Participants were also asked to remember recent critical incidents (Chapanis, 1969) from the previous week.

RESULTS

Users customized software for a variety of reasons, not the least of which was coping with cognitive overload. Giddens (1984) theory of structuration identifies the reciprocal interaction of human actors and structural features of the organization. Figure 1 highlights the main components of this model (arrows a, b, c, d), in which institutional properties and general properties of the technology affect the users' use of the technology, users affect the technology, and the technology affects the institution. This model is then extended to include the effect of individual decisions on others in the group (arrow e), the effect of external events on individual decisions (arrow f), the relationship between individual factors and users' decisions to explicitly customize their software environment (arrows g and h) and the relationship between software manufacturers and the specific technology used within the organization (arrows i and j). Giddens' analysis operates at the organizational level, whereas as this analysis is primarily focused at the individual and group level.

Figure 1: Factors that affect how people adapt technology and how they adapt it for their own purposes (from Mackay, 1990a).

Table 1 summarizes the factors that users cited as both triggers and barriers in their decision-making process about when and how to customize software. The factors are listed from most to least common based on the categories identified in Figure 1, preceded by the percentage of study participants who cited that factor. Note that these data were compiled from open-ended questions and as users explained specific reasons for making or avoiding particular customizations. Participants were not given this list and asked to identify those that were relevant to them.

Participants identified 31 unique triggers (from a total of 226), an average of 4.4 triggers cited per person. All but two participants cited at least one trigger and one person cited 11. Participants identified 20 unique barriers and cited a total of 102, an average of two barriers per person. All but four participants cited at least one barrier and one person cited seven.

Triggers: Several overload situations acted as triggers to customization. Participants were most likely to customize when they discovered that they were doing something repeatedly and chose to automate the process. Equally common was a reaction to a system change, when users would retrofit the software to act as it did prior to the system change. Also very common was customization for the purpose of stopping something that was annoying or slow. (This was often cited in conjunction with the repetition.) Other triggers included discovery of things that no longer worked or trying to create a stable environment to support switching among environments (either among different machines or from home to work). Other triggers that were not associated with coping with cognitive overload included observation of what their colleagues had done or exploring the system when it was new.

Barriers: Of course, people did not always cope with overload situations by customizing. The biggest barrier was lack of time, cited by almost two thirds of the participants. Lack of knowledge about customizing (33%) was also a big barrier. Lack of interest in customizing and the general feeling that a particular problem is not worth fixing were also cited.

Percent of users	Customization Triggers	Percent of users	Customization Barriers
Technology			
29%	Something breaks	33%	Too hard to modify
25%	Learn new system	10%	Poor documentation
25%	Switch environments	6%	New customization format
2%	File system gets full	4%	Unpleasant customization process
2%	Poor documentation	4%	System is too slow
4%	Avoid software to avoid retrofit		
2%	Software too limited		
2%	Too cumbersome to find information		
The Organization			
39%	I see something neat	8%	Use Athena's standard commands
25%	Setup for me when I arrived		
4%	Someone posts an idea		
4%	Make generalizable for others		
2%	My manager suggested it		
External events			
43%	Retrofit when system changed	12%	System upgrade broke things
12%	Change job or activities	4%	Early bad experience
10%	Urgent need	2%	System changes too often
4%	Test new application		
4%	System upgrade		
Individual factors			
43%	Notice my own repeated patterns	63%	Lack of time
41%	When it gets too annoying	12%	I'm not interested
22%	I think of something new	10%	Lack isn't painful enough
18%	Learn from it, curiosity	8%	I'm rooted in my old patterns
16%	I delete when I don't need it	6%	I don't know the possibilities
14%	Aesthetics	6%	I'm afraid to risk it
14%	When I'm bored or waiting	4%	I don't know what I need yet
10%	Whim	2%	I refuse to sanction it
6%	Increase productivity		
6%	It's fun		
6%	I'm bored with current one		
4%	Remove clutter		
4%	My mental timer goes off		
4%	Finally understand a customization		
4%	Increase efficiency		
2%	Tending my personal repertoire		
31	Unique triggers	20	Unique barriers
226	Total responses	102	Total responses
96%	Percent of participants	92%	Percent of participants
4.4	Mean triggers cited per person	2	Mean barriers cited per person

Table 1: Factors cited as triggers and barriers to customization (Mackay, 1991).

One can compare the decisions about learning and customizing a new software package to choosing when to invest in a new, depreciable capital investment. The new software package has a learning curve associated with it, which is the cost of 'buying' it. For the sake of discussion, assume that the user has free choice among a number of available software packages. Each software package 'depreciates' as other more effective packages become available or as new features are added that must be learned. When do users switch? At what point does the cost of learning something new become preferable to using out-of-date software? These data support the idea that users 'satisfice' (Newell & Simon, 1972) rather than optimize. People are busy and switching takes time. So customizing becomes a mechanism for easing the transition between old and new software, a set of trade-offs that must be constantly re-evaluated. With few exceptions, people only customize when it is worth the trouble and they already know how to make the desired changes. Users actively take their work context into account when deciding whether or not to customize. For example, if a manager must produce a report by 5:00 pm, she is likely to avoid investing in creating a routine that automates a repetitive procedure, even if it takes 20 minutes to do the procedure by hand. The latter is annoyingly slow, but predictable, whereas the customization routine is very risky: it may not work at all and even it does, there is little benefit derived from turning in the report by 4:30.

Customizations that allow users to continue working as they did before, without learning new patterns of behavior and customizations that increase efficiency by performing a commonly occurring set of actions with a single command, are the most likely to be considered worthwhile. People were particularly sensitive to external system changes that required them to modify their own "automatic" behavior, such as typing particular keys to perform particular functions. They were must likely to customize by retrofitting the new system to respond like the old. Unless the user was bored or just learning the system, aesthetic or "interesting" customizations were generally avoided.

Patterns of Sharing Customizable Software

Technology developers usually view customizing software as a solitary task. After all, the goal is to allow individuals to express their personal preferences in dealing with the technology. Yet this study indicated that customization is often a highly social phenomenon. Sharing customizations served as an important method of establishing and maintaining standard patterns of behavior throughout the organization. People were clearly overloaded and they looked to their friends and colleagues for help. Borrowing customizations had numerous advantages for individual users. They could reduce the time spent learning how to customize, which increased the time available for accomplishing actual work. They could also experience how other people work, find out new ways of doing things and benefit from each other's innovations.

Figure 2 displays the customization exchange patterns in the organization, just prior to a re-organization. Circles indicate individuals, identified by their job category. For example, "A8" a manager, who was labeled as the eighth person in the Visual Computing Group. Clusters of circles indicate groups within the organization, such as User Services and Administration. Arrows are directional, indicating the source and recipient of customization files. Note some of these exchanges involved givers who were proactive, such as when someone explicitly mailed a customization file to someone else. In other exchanges, the recipient was proactive, such as when someone found a useful customization file by looking in someone else's files. This graph does not indicate the number of exchanges that occurred over time; it simply shows that at least one exchange occurred. Several of these exchanges involved a single file that was popular and copied by several people. For example, D5 had a screen background pattern that many people copied and D8 had a technique for creating multiple collections of windows on the screen. In each case, the person gave only that item to other people.

Figure 2: Network of sharing customizations within the organization (Mackay, 1990b).

Sharing of customization files took two different forms. The first was relatively anonymous, in which the customizations were either broadcast to other members of the staff or the file was placed in an accessible location. People who obtained customizations in this manner had to be proactive and sufficiently skilled to interpret the customizations. The second form involved conversations between people in addition to the exchange of files. In

these situations, one person would attempt to identify the needs of another before suggesting particular customization files or techniques.

Anonymous sharers: These people made their files available to everyone and had little idea who used them (or even if they were useful!). They were almost always technically-skilled programmers who enjoyed pushing the software to its limits and reacted to peer pressure to "do neat stuff". Most had created complex sets of customizations that were of interest primarily to other highly technical people. Because they were the most technical members of the staff, members of this group were usually the first to investigate new software packages and usually set up the default files that would be used by everyone else. This often caused problems, since their instincts about how best to set up the technology often did not match the needs of their less-technical colleagues. These anonymous sharers liked to show off their technical prowess, but often had difficulty communicating directly with the rest of the staff.

Translators: These people liked to help their colleagues by making the software environment easier to use. Although generally less technically skilled than the programmers, they had far better communication skills. They translated complex files into simpler ones in order to provide practical benefits to the recipients. Most of them understood the basic design of the system and could talk to the technical staff or borrow their files if necessary. They were more interested in customizations that solved practical problems than ones that demonstrated technical brilliance. They often viewed their role as trying to protect their colleagues who either did not understand or were simply not interested in learning technical details of the system. Unfortunately, because they were less technically skilled, some of their customization files contained errors, which were unknowingly passed on to their colleagues. Translators appear similar to the "gatekeepers" identified by Allen (1972) or local heroes identified by Nardi and Miller (1989). Gatekeepers are highly-skilled individual contributors in engineering organizations who actively seek technical information outside of the organization. Like translators, they translate the information into a form that is easier for their internal colleagues to use and understand. However, the translators in this organization were unlike gatekeepers in that they were not the most technically-skilled members in the organization. They were closer to the “local heroes” in that they did not fill the role all the time; they only acted as translators when there were people who appeared to need the help. The differences may stem from the contrast between customization activities, which occur only when someone needs help getting set up or reacting to an externally-imposed change, and the on-going need for accessing current technical information.

"Anonymous sharers" appeared to perform their function independently of the needs of the individuals of the organization. They did not react to reorganizations or job changes and

generally broadcast customizations whenever they happened upon something interesting. In contrast, translators were very much affected by their roles in the organization. They were much more likely to remember who they gave files to and why. Most performed the role when the need arose and stopped when circumstances changed. For example, in the video group, D4 willingly gave up the role when D3 arrived. Over the course of the study, each group (except the system programmers) always had a clearly-identifiable person acting as a translator.

Customization and Cognitive Overload

Customizing software was an important mechanism for the members of this organization to deal with the cognitive overload caused by constant technological change. Certain kinds of customization, such as retrofitting a new version of the software to act like the previous, familiar version, proved effective in maintaining a more stable work environment. Similarly, capturing repeated tasks and automating them reduced the overload involved in mechanically repeating the task over and over.

Customization in this organization was a highly social activity. People learned effective work strategies from each other by copying their customizations. In many instances, individuals who moved from one group to another changed their software environments to directly reflect that of the rest of the members of the group. Even though this, in the short term, increased the load on the user by forcing him or her to learn a new way of working, in the long run, it reduced overload by allowing the user to benefit from the collective experience of his or her peers. This implies that research into the cognitive overload syndrome must take social factors into account when trying to provide tools and processes to help manage it.

CONCLUSIONS

If we agree that users co-adapt technology, what would it mean to create software that explicitly supports this process? How can we, in our role as designers, benefit from the tendency of users to re-interpret their technology in the context of their own work and to modify it in ways that we cannot anticipate?

Studies of white collar productivity indicate that the introduction of computers has been correlated with a *decrease* in productivity and a corresponding increase in cognitive overload (Dertouzos et al, 1989). Although computers offer flexibility and power, productivity gains will only be achieved if users *use* the technology effectively. This study provides evidence that users' co-adaptation of technology provides an effective mechanism for addressing cognitive overload. The following design considerations will help users adapt the software to meet their own needs and reduce the level of cognitive overload.

1. *Reflection*: Provide users with feedback about the effectiveness of their use of the technology (including customizations) and provide opportunities to reflect upon their processes of use.
2. *Context*: Allow users to encode patterns of behavior and informally include information about their current work contexts.
3. *Sharing*: Assume technology use is embedded in a social structure and provide mechanisms that support sharing and exchange of software, especially among translators, to encourage innovation and share effective methods of accomplishing tasks.

Development of reflective software

The X-Windows System is extremely poor at providing users with mechanisms for reflecting upon their use of the technology. Hidden customization files did not help; users were left to guess, often incorrectly, how their behavior affected the system's actions. Software manufacturers should consider designing software to be *reflective*. Reflective software is somewhat different from Zuboff's (1988) notion of "informating", which provides users with information about the state of the *system*. Reflective software should increase the user's awareness of their personal use of the software. Techniques used to instrument software for feedback to user interface researchers may be useful here, particularly those that summarize behavior. However, presentation is important: raw keystroke logs are unlikely to help. Since most people do not spend much time evaluating their own patterns of use, these features may be of more help to translators than regular users. However, given the influence of these people on the rest of the organization, simply helping translators may be sufficient to significantly increase the productive use of customization. Users should be able to use these reflections or traces of their behavior in order to organize their own behavior in more productive ways.

Help users capture and customize work context

As in the Information Lens study mentioned in the introduction, some users created multiple versions of certain customizations that could be executed at different times as specified by the user. This provided a mechanism for users to create context-dependent sets of customizations without having to articulate in precise terms the conditions under which each was appropriate. For example, users created different window layouts, with different applications running, for use when performing different jobs.

These studies demonstrate that 1) users find it very useful to organize collections of tasks together that are appropriate in different work contexts and 2) explicitly articulating exactly *what* the context is is not only difficult, but sometimes impossible. Some of the most interesting user innovations provided users with mechanisms for stating these different context-dependent states without making the actual states explicit. Technology developers

should consider how to support this need. The first step is to provide an easy mechanism for identifying patterns of behavior or collections of functions and allowing them to be accessed as a group. The second step is to allow users to run these collections independently, either at the times the user decides are appropriate or when certain events trigger the activity. Users in both studies invented a number of different ways of encoding these patterns and used them extensively.

Provide support for sharing customizations

Users actively exchanged customization files and electronic mail rules with each other. Technology developers should consider the effects of sharing customizations on the use of their software over time. Because most use of customizations is not reflective (i.e. users don't usually examine which customizations are truly effective and which are not), patterns of use do not necessarily improve over time. Software designers should consider how to help the people who produce customizations for their peers effectively share their files. The quality of the examples they create will affect the overall perception and use of the manufacturer's software. Specific decisions about the design of customization features also significantly affects how the software is used. For example, providing a customization capability via a "direct manipulation" interface, with no accessible record of it, may make it easier to modify individual features but harder to share them, because users might not understand the form in which the customizations are stored nor be able to gain access to them. Users need to be able to borrow and apply patterns of behavior created by others, and to modify them for their own purposes.

Co-Adaptation

This study involved detailed observations of users who found themselves in a state of cognitive overload. This research illustrates ways in which people have successfully co-adapted, adjusting their own behavior to more effectively use the technology (i.e., adapting to it), and at the same time, re-interpreting and changing the technology to meet their current needs (i.e. adapting it). Participants in these studies reported how both strategies were able to help them reduce their perceived state of overload and help develop more effective ways of working. Technology designers should consider first grounding design in existing, successful work practices, and then exploring how to augment those work practices with new technology, under the user's on-going control. This approach permits users to reflect upon their work activities, identify situations of overload, and explicitly adapt the technology for their own purposes, with the aid of their colleagues, in order to reduce cognitive overload and help manage the complexity of the workplace.

REFERENCES

Allen, T.J. (1972). Communication Networks in R&D Laboratories. *R&D Management*, 14-21.

Barley, S. R. (1986). Technology as an Occasion for Structuring Evidence from Observations of CT Scanners and the Social Order of Radiology Departments. *Administrative Science Quarterly*, 31, 77-108.

Champine, G. (1987). Project Athena as a Next Generation Educational Computing System. *ASEE Annual Conference Proceedings.* ASEE.

Chapanis, A. (1969). *Research Techniques in Human Engineering*. Baltimore, Maryland: John Hopkins Press.

Dertouzos, M., Lester, R. and Solow, R. (1989). *Made in America: Regaining the Productive Edge*. Cambridge, Massachusetts: The MIT Press.

Eveland, J. and Bikson, T. (September 1988). Work Group Structures and Computer Support: A Field Experiment. *Proceedings on the Conference for Computer-Supported Cooperative Work*, 39-51. Portland, Oregon.

Giddens, A. (1984). *The Constitution of Society: Outline of the Theory of Structure.* Berkeley, California: University of California Press.

Lampe, D.R. (February 1988). The MIT X Consortium. *The MIT Report*.

Landauer, T. (1986) *The Trouble with Computers*, Cambridge, MA: MIT Press.

Lovelock, J.E. (1979). *Gaia: A New Look at Life on Earth*. Oxford, England: Oxford University Press.

Mackay, W.E. (1988). Diversity in the Use of Electronic Mail: A Preliminary Inquiry. *ACM Transactions on Office Information Systems*, 6(4).

Mackay, W.E. (1990a). *Users and Customizable Software: A Co-Adaptive Phenomenon*, Massachusetts Instititute of Technology.

Mackay, W.E. (1990b). Patterns of Sharing Customizable Software. In *Proceedings of ACM CSCW '90: Conference on Computer-Supported Cooperative Work*. Los Angeles, California: ACM.

Mackay, W.E. (1991) Triggers and barriers to customizing software. In *Proceedings of ACM CHI '91 Human Factors in Computing Systems.* New Orleans, Louisianna: ACM/SIGCHI.

Margulis, L. (1986) *Microcosmos.* NJ: Summit Books.

Nardi, B. and Miller, J. (October 1990). Twinkling lights and nested loops: Distributed problem solving and spreadsheet development. *Conference on Computer-Supported Cooperative Work.* Los Angeles, California: ACM.

Newell, A. and Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Norman, D.A. (November 1981). The Trouble With Unix: The User Interface is Horrid. *Datamation,* 139-150.

Orlikowski, W. (1992). *The Duality of Technology: Rethinking the concept of technology in organizations. Organization Science.* 3(3), pp. 398-427.

Sproull, L. & Kiesler, S. (1986). Reducing Social Context Cues: Electronic Mail in Organizational Communication. *Management Science,* 32(11), 1492-1512.

Suchman, L. (1987). *Plans and Situated Actions*. Cambridge, England: Cambridge University Press.

Suchman, L. and Wynn, E. (1984) Procedures and problems in the office. *Office: Technology and People.* Vol. 2, pp. 133-154.

Turkle, S. (1984). *The Second Self*. New York, New York: Simon and Schuster.

von Hippel, E. (1988). *The Sources of Innovation*. New York: Oxford University Press.

Zuboff, S. (1988). *In the Age of the Smart Machine*. New York: Basic Books.