Patterns of Sharing Customizable Software

Wendy E. Mackay

Massachusetts Institute of Technology
E40-366
1 Amherst Street
Cambridge, MA 02139

Abstract

The act of customizing software is generally viewed as a solitary activity that allows users to express individual preferences. In this study, users at two different research sites, working with two different kinds of customizable software, were found to actively share their customization files with each other. This sharing allowed the members of each organization to establish and perpetuate informally-defined norms of behavior.

A small percentage of people within the organization were responsible for most of the sharing. One group of these were highly-skilled software engineers, who were usually the first to try new software. They used customization as a way to experiment with and learn about the software and made their files available to others through various broadcast mechanisms. This group did not try to determine whether their customizations were useful to other users. The second group were less skilled technically but much more interested in interpreting the needs of their colleagues and creating customization files tailored to those needs. They acted as *translators* between the highly technical group and the rest of the organization.

The spontaneous sharing of customization files within an organization has implications for both organizations and for software designers. Managers should 1) recognize and support the role of translators, 2) recognize that not all sharing is beneficial, and 3) provide opportunities for the exchange of customization files and innovations among members of the organization. Software designers should 1) provide tools that allow users to evaluate the effectiveness of their customizations through *reflective software,* 2) provide well-tested examples of customization files with the first release of the software, 3) explicitly support sharing of customizations, and 4) provide tools to support the activities of translators.

Introduction

Much of the research in cooperative work develops or examines software that is explicitly designed to support information sharing, such as electronic mail (Malone et al., 1987, Borenstein and Thyberg, 1988), group-oriented decision tools (Stefik et al., 1987), project management tools (Begeman et al., 1986, Sathi et al., 1986), and electronic calendars (Greif and Sarin, 1986). However, other types of information sharing occur naturally within organizations, sometimes in unexpected ways. For example, Nardi and Miller (1990) describe the extensive sharing of spreadsheets within organizations. The exchange of customization files provides another example of how users in an organization share information about their preferred ways of interacting with software.

The act of customizing one's own software applications is usually viewed as a reasonably solitary task. After all, what an individual does in the privacy of his or her office, for reasons of personal taste and efficiency, appears to have little to do with the rest of the organization. However, because people have varying levels of desire and ability to customize, and have limited amounts of time, they often look to friends and colleagues for customization ideas. Borrowing customizations has numerous advantages for individual users. They can reduce both the time spent learning how to customize and the risk of making errors, which increases the time available for accomplishing actual work. They can also experience how other people work, find out out new ways of doing things and benefit from each other's innovations.

From a research standpoint, customization files are interesting to study because the users' patterns of use are encoded as artifacts that can be traced throughout an organization, rather like tracing the spread of archeological artifacts throughout a population. More importantly, the use of these files continues to affect daily patterns of behavior over time. Systematic study of records of changes, correlated across the individuals in an organization, can provide new views of the communication networks that evolve within the organization.

The purpose of this article is to discuss the social aspects of an apparently non-social practice, that of customizing one's software environment. The study was conducted at two research sites, with two different forms of customizable software. The first is the Information Lens (Malone et al., 1987), which helps people customize the process of managing their electronic mail. The second is a set of software for the Unix operating system, using the X Window System. The associated customization files are designed to let people express preferences about how they set up and use software applications.

Exchange of Information Lens Rules

The first study was conducted as part of a study of the Information Lens, which is designed to help users filter and organize their electronic mail. The Information Lens can be viewed as a method for customizing the process of managing one's mail with IF-THEN rules. The rules usually identify text strings in the TO:, CC:, FROM:, and SUBJECT: fields of a message and perform operations such as deleting the message or moving it to a specified mail folder.

Research Site

The research site is a large laboratory with approximately 60 people within a research center at a major American corporation. The site was chosen because of its use of the hardware and software environment necessary for the operation of the Lens prototype and its extensive use of electronic mail. The site and the original study are described in greater detail elsewhere (Mackay, 1988, Mackay, 1989).

Participants

Eighteen members of the laboratory used the Lens software over a period of three or more months. 15 are full-time researchers and three are managers. Of the researchers, six are computer scientists and nine are trained in physics, psychology, anthropology, or sociology.

Data

The data include a series of open-ended interviews about the use of electronic mail and the design of Lens rules. Interviews were conducted prior to participation in the study and at

two-to-three month intervals during the course of the study. In addition, automatically-generated data were collected, including snapshots of each user's rules, taken either weekly or whenever the any of the Lens functions were used.

Results

Four people shared rules or information about rules with others in the organization. The first two are not surprising. I, as the researcher, provided five sample rules for beginning users. Three of these rules were chosen because users in an earlier study had found them useful and the other two rules illustrated a trick that solved a commonly-encountered problem. The second person who shared rules was the developer of the local version of the software. One might have expected him to be the primary source of rules because of his extensive knowledge of the software. But in fact, although he gave rule advice to approximately one third of the active Lens users, he rarely gave them actual rules. His own personal rules were designed to test the system and were quite different from those of a normal user.

Two regular Lens users also shared their rules with other members of the organization. One of them could be considered a "rule-guru": a highly technical person who wrote the most rules and provided feedback and ideas to the developers. Considered one of the top technical members of the lab, he began experimenting with Lens as soon as it became available and created almost 100 rules. He also developed several technical innovations that were later incorporated into the local version of the software.

The rule guru created several complex rules that could be traced to other people's rulesets. I traced three of these rules or rule combinations: One allows users to identify sets of related but uninteresting messages and delete them (called the "boring" rule). The second consists of a pair of rules that saves jokes from a well-liked author and deletes jokes by others. The third rule identifies a person from outside the lab who distributes unwanted seminar announcements. This rule deletes the announcements without deleting personal messages from the author. (Most people were uncomfortable with automatically deleting messages from a particular sender, even if the sender had never sent anything of interest.)

Four people copied the boring rule, and three each used the pair of joke rules and the delete rule. The people who copied these rules are all programmers with from 6 to 62 of their own rules. One person adopted the rule-guru's entire set of almost 100 rules. This colleague quickly found them inappropriate and stopped using Lens for many months. He then rediscovered its usefulness when filtering mail after returning from a trip.

The fourth person who shared rules had no computer training, but liked and used the locally-developed version of Lens. He experimented with a set of Lens rules, based on a strategy, developed by a colleague. He then provided a complete copy of his rules to another non-technical colleague. He was able to successfully translate from the more complex version used by the technical staff to a less complex version that directly met the needs of a fellow staff member with needs similar to his own.

Another form of rule-sharing also occurred. Two manager-secretary teams agreed on a standard form of communication and created a corresponding set of rules to facilitate processing of the mail messages. In another case, members of a group agreed to customize their Lens rules to make their interactions with each other more efficient. These examples involve a social commitment to work together in a particular way instead of simply sharing individual patterns of working.

The function served by these rule sharers appears similar but not identical to the gatekeepers identified by Allen (1972). Gatekeepers are skilled individual contributors

who actively seek technical information from outside of the organization and translate it for the use of their colleagues. Here, several people within the organization examined a new software application and created forms of it specifically tailored to the needs of others in their group.

These findings raise several questions. What are the characteristics of people who provide customizations for others? Are customization exchanges one-or two-way? What are the circumstances under which people decide to share their ideas and customizations? The second study was designed to investigate these questions further, as part of a larger study of how users customize software.

Sharing of Unix Customization Files

Research Site

Project Athena was created as an experiment in educational computing at MIT, sponsored by Digital and IBM. (Balkovich et al., 1985, Champine, 1987). Project Athena has over 1,000 Digital and IBM workstations available 24 hours a day to students, faculty and staff and is the largest centrally-managed distributed computing system in the world. The site and the original study are described in greater detail elsewhere (Mackay, 1990).

Project Athena's computational environment is based on B4.3 Unix and the X Window System. Users have a variety of choices about text editors, formatters, window managers, mail systems and other application software. Users can specify how each application looks (e.g. font sizes, borders, colors, shapes) and how to interact with it (mouse, key bindings, menus, etc.). Users also have a variety of customization options. Some affect all applications, while others are specific to a particular application.

Participants

The Project Athena staff includes over 80 people, including managers, secretaries, technical and non-technical staff. They provide a variety of services, similar to the MIS department of a large corporation. The major groups include: hardware and software operations, systems development, external relations, internal relations, finance, personnel, user services (documentation, consulting, training, user accounts), and administration. Digital and IBM also maintain technical staff on site. Note that this organization has a very real customer base and real deadlines: a software company may "slip" a release date, but MIT never slips the beginning of the semester. An error in the release may affect thousands of people. 51 staff members volunteered to participate in the study, representing a cross-section of the staff, including manager, secretaries, systems programmers, applications programmers and other non-programming staff.

Table One summarizes the technical background of the members of the organization. A rating of "high" indicates an advanced degree in computer science and five or more years of computer programming experience. A rating of "medium" indicates 3-5 courses in computer science and some application programming experience. A rating of "low" indicates one or no programming courses and no programming experience. Two of the systems programmers did not have an advanced degree in computer science but had over fifteen years of systems programming experience, and so were given a "high" technical rating. Note that only two of the applications programmers have a "high" technical rating and three have a "low" rating. The majority of the rest of the staff have a "low" technical rating.

Table One: Technical Level of Staff Members

N	Job Category	High	Medium	Low
10	Manager	3	2	5
5	Secretary	0	0	5
12	Sys. Programmer	12	0	0
8	App.Programmer	2	3	3
16	Other staff	2	4	10
51	Total	19	9	23

Data

I conducted one or more open-ended interviews in each participant's office, over a period of four months. Participants described their customizations (and innovations) and identified the sources and recipients of their customization files. Participants also rated their frequency of communication with each person on the staff and filled out a questionnaire about their work background, technical skills, the software they use and how they customize it. I obtained copies of the update history for each person's workstation, at least two snapshots of their customization files and change dates, and copies of selected customization files.

Results

Over three-quarters of the participants stated that they had received customization files from other people when when they joined Athena, in addition to the system default files. Newcomers, technical or otherwise, tend to ask for help when they arrive, not only to get examples that have proven to be useful, but also to make social contacts. Staff members identified the following methods by which they had obtained or given customizations:

1. Someone helps you get set up.
2. You ask someone to help you get set up.
3. You get the standard system file and use it.
4. You have a problem and ask someone for help.
5. New ideas are posted electronically in a common area and you look.
6. Someone has a new idea and tells you about it.
7. Someone tells you to look in the common area.
8. You have a symbolic link to someone else's file which is automatically updated.
9. You walk by and see someone else's screen and ask how something was done.
10. You watch someone performing some task, notice a useful technique, and ask how it's done.
11. You help a newcomer get set up with a version of your files.
12. You post an idea in the common area.
13. You tell your friends about a new idea.

Project Athena's default customization files vary in complexity. Some provide all possible customization options. A small number of highly skilled users find this useful, especially when they are interested in seeing a working example of a specific option. Most other users, who are not only less experienced but also less interested in the software, find these files overwhelming. They prefer to use more limited files with customizations that had already proven useful for someone with similar job responsibilities and work patterns.

Some users are proactive about learning how to customize, others are reactive and customize only when someone offers to help. Most people (78%) made some customizations, although only 11% took the time to completely and systematically customize their work environments. All of the latter were formerly or currently programmers. Some of these programmers create files for general use and publicize their customizations electronically, thus sharing their files with the rest of the organization. However, most other staff members find these files difficult to use. Interestingly, the most senior of the technical staff choose to customize very little, if at all. Having experienced many systems changes over the years, they are no longer interested in "playing" with the system and only create customizations that facilitate their immediate work. A number of non-technical staff members never create their own customizations and ask for help if they become stuck. In general, most people described themselves as "too busy" or not knowledgeable enough to fully customize their software environment. One moderately technical staff member described the process as follows: "I try out a new application and decide if it's useful or not. If it is, I play with it for a while and make customizations. After that, customizations are very rare."

Given these barriers to customization, it is not surprising that users often choose to get information about customization from each other. Participants were asked to rate the different sources of information about customization, where 0 = "never use", 1 = "use rarely", 2 = "use sometimes" and 3 = "use often". The means for each group are listed in Table Two. The columns are ordered from highest overall rating on the left to lowest on the right.

Table Two: Mean Ratings for Sources of Information about Customization

N	Job Category	Ask a person	Copy & experiment	Manual pages	Documen-tation	Read source	Write own
10	Manager	2.5	2.5	2.4	1.9	0.7	0.8
5	Secretary	2.6	2.2	1.2	1.8	0.0	0.0
12	Sys. Programmer	1.8	2.1	2.4	1.3	2.3	2.0
8	App.Programmer	2.4	2.4	2.4	1.6	1.1	1.1
16	Other staff	2.7	2.2	2.1	1.8	1.0	1.1
51	Total	2.4	2.3	2.2	1.7	1.2	1.1

Scale: 0 = never use 1 = use rarely 2 = use sometimes 3 = use often

People in all job categories except systems programmers, preferred to "ask a person", followed by "copy and experiment" (which involves borrowing someone else's file and editing it). Reading the source code and writing their own version of the code from scratch were considered the least useful. Systems programmers were almost opposite, preferring to read the source code and look at the on-line manual pages to asking their

colleagues. This was true even when they had no specific technical knowledge of the software.

Figure 1 shows a diagram of the patterns of sharing of customization files among Athena staff members. The codes in ovals represent individuals, identified by job category, where A = manager, B = secretary, C = systems programmer, D = applications programmer, and E = other staff. The arrows between the ovals indicate that one or more customization files were exchanged between the two individuals. The direction of the arrow indicates who was the source and who was the recipient of the customization file. Double-headed arrows indicate an exchange of customization files.

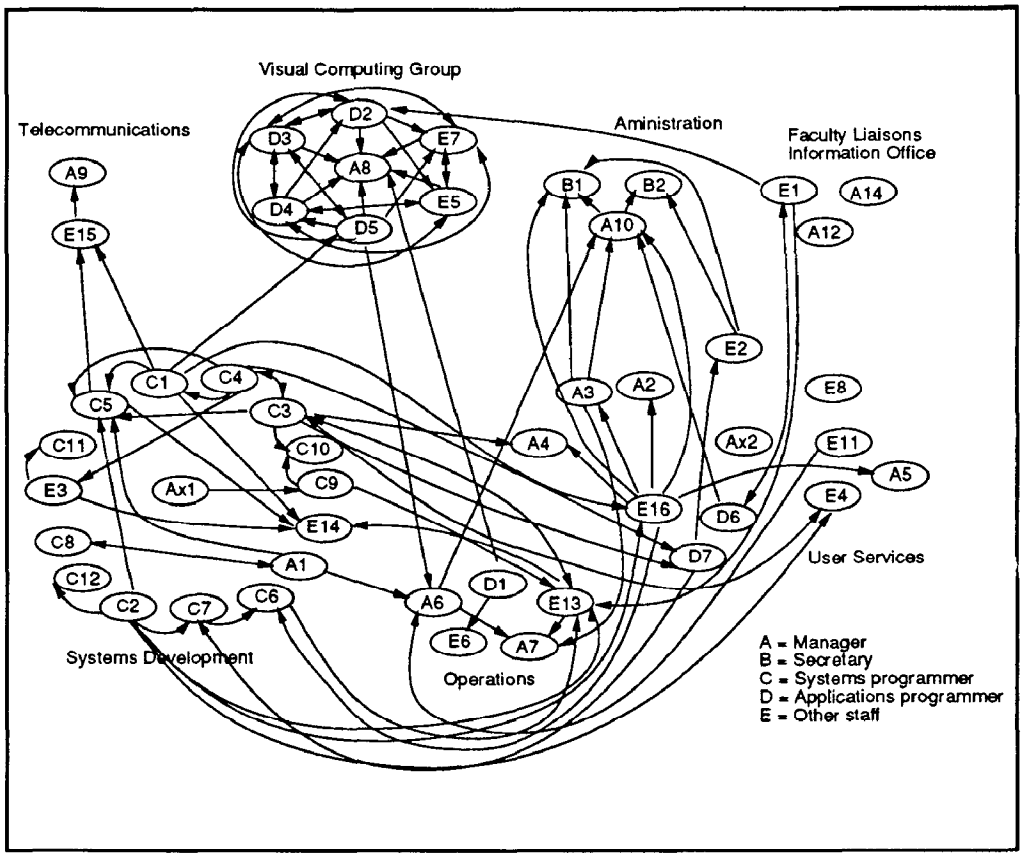

Figure 1: Exchange of Customization Files at Project Athena

Members of the video group actively exchange information with each other, but rarely share their files with others outside the group. Most of the members of the group have developed at least one customization that others have borrowed. The most active sharer was formerly D4 and is now D3. The administration group is the least technical of all of the groups. They receive customization files from outside the group, but rarely give files to others outside. Here, A10 is the most active, bringing in files from the outside and giving them to the members of the group. E2 and A3 have also given their files to members of this group. The remaining groups share as often with members of outside groups as within their own groups. This is due partly to a recent reorganization, which affected the majority of the Athena staff.

Table Three presents a different measure of sharing. Participants were asked how many people they borrowed customization files from and how many they lent customization

files to. Columns Three and Four show the mean number of people with whom the staff member has borrowed or lent files. Not surprisingly, the least technical staff members borrow from the most people and the most technical staff members borrow the least. However, the systems programmers do not lend to the most people. Instead, the applications programmers, who represent a mix of technical expertise, lend files to the most people.

Table Three: Sharing Files

N	Job Category	Borrow Files	Lend Files	Same Layout	Open Files	Trans-lators
10	Manager	2.6	1.4	80%	20%	20%
5	Secretary	3.2	0.6	80%	0%	0%
12	Sys. Programmer	1.8	2.3	92%	67%	8%
8	App.Programmer	1.9	3.8	50%	63%	38%
16	Other staff	2.3	2.1	81%	50%	25%
51	Total	2.3	2.0	78%	45%	20%

An interesting indication of the extensive level of sharing is the use of a common screen layout. Column Five in Table Three shows that 78% of the staff members used the same basic layout. Seven of the nine people who created unique screen layouts were from the video group, which is the most isolated of the groups within Athena. Members of the staff have the option of making their files "open" or freely accessible to others to copy and use. Column Six shows that 45% make their files accessible in this way. Several of the systems programmers commented that they found this an especially useful way of sharing files because it does not require them to talk to the recipients of the files.

A few people actively share their files and talk directly to the recipients of the files. These people have been designated as *translators*. Column Seven lists the percentage of people in each group who act as translators. Note that the systems programmers, who are the most likely to make their files accessible to others are among the least likely to act as translators. Managers rarely act as translators. Of the two who do, one does it to protect her group from an ineffective translator and the other does it in lieu of some of her other job responsibilities (and has been down-graded as a result.)

At the time of the study, each group had a single, self-appointed translator. Although several people in each group expressed an interest in this role, only one person acted as a translator at a time. However, two people acted as "translators-at-large" for the entire organization. Table Four shows the translators identified for each group, both prior to the study and at the time of the study. Managers are coded with an A, secretaries with a B, systems programmers with a C, applications programmers with a D, and other staff members with an E. These people are notable in Figure One because they exchange files with more people in the organization.

Note that systems programmers (code C) do not act as translators, nor do they need or want someone in their group performing this function for them. None of the secretaries (code B) have sufficient technical skills to act as translators for the others. In the Administration group, E2 (a non-programmer) originally provided the files for secretaries. As mentioned earlier, the manager (A10) took over the translation role herself after she

discovered that E2's files contained errors, were confusing and made their software environments non-standard. She told her staff to ignore the contributions from E2 but did not tell E2 that his files were no longer welcome.

Table Four: Translators within each group

Group	Before Study	During study
Administration	E2	A10
Video group	D3	D4
Operations (hardware)	D1	E13
Operations (software)	E13	E13
User Services	E16	E16
Education Initiatives	E1	E1
Systems Development	none	none
Works with several groups	E16 A3	E16 A3

An applications programmer in the video group, D3, acted as a translator for several years. When D4 arrived and showed an interest in helping the other members of the group, D3 stopped acting as a translator. He felt it was convenient to stop this role "when someone else appeared on the scene to do it...we don't need two people to do it". He described the group's attitude: "If there's a vacuum, someone steps in to fill it." The situation was somewhat different in the operations groups. D1 performed this role for the hardware group and was then promoted to an applications programmer position. He became the least technical member of his new group and stopped acting as a translator. E13, who acted as a translator for the software operations group, expanded his role to include the hardware group as well when D1 left. E16 and A3 act as translators for people throughout the organization.

A common impetus for sharing, particularly by translators, is any change in the software environment. Many users come to rely upon their customizations and actively resist situations that force them to change their behavior. This was apparent when the standard window manager for Project Athena changed as part of a regularly-scheduled software release. All users (except a few senior technical staff) received it automatically. Several staff members deeply resented this change. One said, "I hate having MWM [the new window manager] forced on me -- I hunt it down and kill it with extreme prejudice". This staff member is the translator for his group and sent an electronic mail message with instructions on how to revert back to the old window manager. He later sent out instructions for how to incorporate customizations from the old window manager to make the new one act like the old one.

Only 40 of the 51 participants in the study were faced with a change from the previous window manager to the new one. (The others were new to the organization and were given the new window manager as soon as they arrived.) Of these, 78% found a way to maintain a stable user interface, either by retrofitting their existing customizations from the old window manager or setting a variable to enable them to continue using the old window manager (shown in Table Five). *All* of the people who kept their interactions the same had already customized their software in some way. The remaining 22% of the staff changed their interaction patterns by learning the new window manager when it appeared. *None* of these people had customized their previous window manager. The members of this group gave several reasons for learning the new software rather than continuing to use the software they were accustomed to. Seven of the non-technical staff

members were unaware that they could revert to the old software. Faced with a completely new interface, each expressed dismay, but didn't realize they had the option of changing it. Others explicitly chose to use the standard Athena software. Still others simply refused to customize, even if it meant extra time in learning a new user interface.

Table Five: Resistance to Change

N	Job Category	Retrofit old Window Manager	Refuse new Window Manager	Total Percent who stay the same
8	Manager	63%	0%	63%
1	Secretary	0%	0%	0%
11	Sys. Programmer	55%	27%	82%
7	App.Programmer	71%	14%	85%
13	Other staff	62%	23%	85%
40	Total	60%	18%	78%

Summary and Conclusions

An important property of any user interface is that it both guides and constrains the patterns of interaction between the user and the software application. Increasingly, application software is designed to be "customizable" by the end user, providing specific mechanisms by which users may specify individual preferences about the software and how they will interact with it over multiple sessions. Users may thus encode and preserve their preferred patterns of use. These customizations, together with choices about which applications to use, make up the unique "software environment" for each individual user.

In this study, users actively shared customization files with each other. Users have a number of incentives to share customizations, including taking advantage of each other's work, learning new methods of accomplishing useful tasks, avoiding errors, and generally saving time. The net result is that users adopt patterns of use from each other, propagating both useful innovations and errors throughout the organization.

Users are most likely to spend time customizing when they first learn a new software application, which is also when they have the least knowledge about the software and their eventual use of it. They are also most likely to borrow customizations at this time and adopt other people's patterns of working, regardless of whether those patterns are effective. Once users invest time in developing and learning a set of customizations, they will attempt to maintain those customizations over time. Thus, software upgrades that change the user interface are likely to cause users to spend time customizing some or all of the new interface to act like the old.

Some of the people who share customizations are highly-technical programmers, who experiment with the software and then make their first set of customizations available to others in the organization. These files are often complex and rarely reflect the needs of other users in the organization. Despite this, many people adopt these files to avoid learning how to make their own customizations.

A second group of customization sharers act as translators, creating simplified and more task-specific sets of customizations. These people often base their work on the files created by the technical experts. Unlike the first group, who usually share their files through various broadcast mechanisms, they enjoy talking directly to their colleagues and get satisfaction from helping to make their colleagues' lives easier. Although most are not trained programmers, most translators understand the basic design of the software. As one said, "In X, you learn that everything visible is probably customizable...After you know that, you just have to decide to figure out what something is called and then where to change it." Translators are interested in customizations that solve practical problems rather than those that demonstrate technical skill. They try to protect those who either don't understand or are simply not interested in learning more about the software. Unfortunately, because translators rarely have technical training, their customization files are more likely to contain errors. Since translators are also the most likely to provide files for newcomers to the organization, inefficient or buggy files may easily be propagated throughout the organization.

The sharing of customization files has implications for both organizations and for software designers. In this study, few of the managers were aware that customization files were being exchanged and none were aware of the extent of sharing. Staff members are not rewarded for sharing and in some cases are down-graded for it because it distracts them from their daily work. Yet, members of the organization clearly feel a strong need to borrow working examples of customization files from each other. Staff members usually appreciate the files created by translators. However, files created by two translators created problems for the recipients. It is not clear how to ensure that files from translators are effective, unless the software provides a mechanism for evaluating the effectiveness of particular customizations. In many organizations, users take software training courses once, prior to or when just learning a new software application. Providing periodic discussions for group members with similar uses of the software would help users to exchange effective customizations, learn new features, provide feedback to people who act as translators and provide a mechanism for identifying problems and errors.

The participants in this study requested the following in the design of customizable software:

1. The ability to browse through others' useful ideas,
2. Better mechanisms for sharing customizations,
3. Methods of finding out which customizations are used and effective, and
4. Methods of identifying customizations that are ineffective.

These suggest several design implications. Software manufacturers should consider designing software to be reflective, allowing users to become more aware of their own patterns of use and providing methods for evaluating effectiveness. 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 how *they* actually use the software. Techniques used to instrument software for feedback to user interface researchers may be useful here, particularly those that summarize behavior. (Raw data, such as 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 the translators than regular users. However, given the influence of these people on the rest of the organization, it is important to help them reduce the number of inefficient or ineffective customizations and increase the creation and sharing of user innovations.

Software manufacturers should also consider the impact of delivering a poorly-conceived set of default values when the first version of the software is shipped. Unlike many features that can be fixed in subsequent updates, decisions that affect individual patterns of use are likely to have long-term effects. Many of the users in this study insisted on using the first pattern they learned when they arrived at the site, and a number spent a significant amount of time retrofitting new "improved" software to be like the old familiar software. The solution is *not* to release software without any examples of customization. This simply shifts the burden of creating sharable customizations to volunteers in the organization, who may lack both technical skills and an appreciation for the elements of a good user interface. Even though some people will continue to improve their customizations, these improvements are less likely to be shared among members of the organization.

The prevalence of sharing within this organization, particularly in the Athena study, suggests that software designers should consider providing explicit mechanisms for sharing customizations within an organization. This may affect the choice of the customization mechanisms provided to users. For example, some kinds of direct manipulation interfaces may be easier to use but harder to share. Finally, software designers should consider how to assist the people who become the translators or sources of customizations for their peers. The quality of the customizations the translators create will affect the overall perception and use of the manufacturer's software.

In conclusion, this study has demonstrated that customization is not a purely individual activity. Members of an organization may actively share customization files with each other and affect each others' behavior for long periods of time, perhaps years. This sharing of customization files may serve to establish and perpetuate standard patterns of behavior throughout the organization.

References

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

Balkovich, E., Lerman, S. & Parmelee, R.P. (1985). Computing in Higher Education: The Athena Experience. Communications of the ACM, 11(28), p.p. 112-124.

Begeman, M., Cook, P., Ellis, S., Graf, M., Rein, G., and Smith, T. (December 1986). Project Nick: Meetings Augmentation and Analysis. Proceedings on the Conference for Computer-Supported Cooperative Work. Austin, Texas.

Borenstein, N.S. & Thyberg, C.A. (September 1988). Cooperative Work in the Andrew Message System. Proceedings on the Conference for Computer-Supported Cooperative Work. Portland, Oregon.

Champine, G. (1987). Project Athena as a Next Generation Educational Computing System.

Greif, I. and Sarin, S. (December 1986). Data Sharing in Group Work. Proceedings on the Conference for Computer-Supported Cooperative Work. Austin, Texas.

Mackay, W.E. (September 1988). More than Just a Communication System: Diversity in the Use of Electronic Mail. Conference on Computer-Supported Cooperative Work. Portland, Oregon: ACM.

Mackay, W.E. (May 1989). Tools for Supporting Cooperative Work Near and Far: Highlights from the CSCW Conference. CHI '89 Conference on Human-Computer Interaction. Austin, Texas: ACM/SIGCHI, Panel presentation.

Mackay, W.E. (May 1990). Users and Customizable Software: A Co-Adaptive Phenomenon. Doctoral Dissertation, Sloan School of Management, Massachusetts Institute of Technology.

Malone, T.W., Grant, K.R., Turbak, R.A., Brobst, S.A., & Cohen, M.D. (1987). Intelligent Information-Sharing Systems. Communications of the ACM, 30, 484-497.

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.

Sathi, A., Morton, T., and Roth, S. (Winter 1986). Callisto: An Intelligent Project Management System. The AI Magazine.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., & Suchman, L. (1987). Beyond the chalkboard: using computers to support collaboration and problem solving in meetings. Communications of the ACM, 30, 32-47.

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