Triggers and Barriers to Customizing Software

Wendy E. Mackay

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

ABSTRACT

One of the properties of a user interface is that it both guides and constrains the patterns of interaction between the user and the software application. Application software is increasingly 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.

While it is theoretically possible for each user to carefully evaluate and optimize each possible customization option, this study suggests that most people do not. In fact, since time spent customizing is time spent not working, many people do not take advantage of the customization features at all. I studied the customization behavior of 51 users of a Unix software environment, over a period of four months. This paper describes the process by which users decide to customize and examines the factors that influence when and how users make those decisions. These findings have implications for both the design of software and the integration of new software into an organization.

KEYWORDS: Customization, Tailorability, Unix

INTRODUCTION

Software manufacturers are beginning to provide users with a greater range of customization capabilities. Yet we still know very little about how users actually customize. A few recent studies have begun to explore how users customize or tailor different kinds of software (Trigg et al., 1987, MacLean et al., 1990, Nardi and Miller, 1990). However, a number of questions remain. How do people decide how much to customize? At one extreme, users may choose to never spend time customizing. They avoid risks and save time, but never obtain any benefits from customization. At the other extreme, users who spend all of their time customizing software will never accomplish any work and, again, no benefits will accrue from customization. How then do users decide what to customize and when? Are there barriers that reduce the likelihood of customization? If so, are they related to features of the software, characteristics of individual users, social factors, external factors or combinations of the above? Are there factors that trigger users to customize? If so, what are they and can they be used to encourage productive customization?

This paper describes the decision process users use when choosing to customize software and identifies the factors that influence their decisions. Customizable software is defined here as having mechanisms that allow users to customize their personal software environment without writing code, with changes that persist across sessions. I collected data about the customization activities of over 50 staff members over a period of four months. A full description of the research study and a more subtle definition of customization can be found in Mackay (1990a).

RESEARCH STUDY

Research Site

MIT's Project Athena is an eight-year, $100 million "experiment in educational computing" sponsored jointly by Digital Equipment Corporation and IBM. The goal was to provide the members of the MIT community with a centrally-managed distributed network of workstations in which users can access their files from any of the 1100 Athena workstations on the campus. (Champine, 1987) All members of the Athena staff, from secretaries to managers to programmers have at least one high-performance workstation with a minimum of 6 megabytes of memory and a 70 megabyte local hard disk, with a mouse and keyboard for input.

Project Athena's software environment is based on a version of the Berkeley Unix operating system. Users may choose from a variety of text editors, text formatters, window managers, communication systems and other software. Each user may choose how much or how little to customize each application and the software environment as a whole. Some applications are difficult to learn, but provide the greatest amount of power, flexibility and customization options. Others are designed to be easy to use but offer limited customization options. A user may try one application for a while and then try another.

Users may express preferences via different mechanisms at different levels. The user can specify how an application looks (e.g. font sizes, borders, colors, shapes), how to interact with it (e.g. mouse, key bindings, menus) as well as preserve preferred patterns of use. Some choices affect all applications, such as the choice of a window manager or the use of Xresources. Others are specific to an application.

Table 1 lists some of the options users have when customizing software in the Athena environment:

Table 1: Customization options

Menus	Screen background	Window manager	Key bindings
Fonts	Screen layout	Window layout	Mouse buttons
Colors	Screen saver	Window size	Icon behavior
Aliases	Information access	Window looks	Status indicators

Participants

Project Athena is run by a group of over 80 people, including managers, secretaries, technical and non-technical staff. The staff members are lead users of a highly-customizable software environment, a distributed network of Unix workstations using the X Window System. They provide a variety of services, similar to the MIS department of a large corporation, including: hardware and software operations, systems development and third-party vendor support, public relations, training, documentation, consulting, faculty liaison, financial and administrative staff, the visual computing group and staff from IBM and Digital. Although located at a University, Athena staff have very real responsibilities and deadlines. A software company may "slip" a software release date, but MIT never slips the beginning of the semester.

51 members of the Athena staff completed the interviews and supplied all of the requested data. (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.) The study participants include a cross-section of managers, administrative personnel and both technical and non-technical individual contributors. Table 2 lists the numbers of participants from each job category and their technical skill levels.

Table 2: Technical background of the study participants

Job Category	N	Technical Level High	Medium	Low
Managers	10	3	2	5
Secretaries	5	0	0	5
Systems Programmers	12	12	0	0
Applications Programmers	8	2	3	3
Other staff	16	2	4	10
Total	51	2	4	10

Technical skill is defined operationally, based on a combination of computer education and programming experience. Individuals rated at a "high" technical level have an undergraduate major or graduate education in computer science and four or more years of experience as a systems programmer. (Two individuals who were not computer science majors but who have been programming as systems programmers for over ten years were also included in this category.) Individuals rated at a "medium" technical level have completed two to five computer science courses and have up to four years of job experience at the applications programmer level. Individuals rated at a "low" technical level have at most one computer course and no job experience as a programmer.

Data

Data was collected over a period of four months, during which a number of changes occurred, including a reorganization of over 80% of the staff, a major office move involving over half of the staff, and a major software upgrade. The data consist of open-ended interviews, questionnaires, and automatic records of customization activities. I conducted the interviews over a period of four months. Prior to each interview, I asked participants 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. I also asked participants to fill out two additional questionnaires, during or after the interviews, which included usage and customization levels of different applications.

Interviews were approximately one hour and conducted in each participant's office to facilitate asking questions about particular files or customization activities. I asked each participant to print out a second copy of the ordered list of customization files, which provides an indication of the user's rate of customizing and identifies which files have been changed. I then asked participants to show me 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. I asked users to remember recent critical incidents of the previous week. This strategy is based on the Critical Incident Technique. (See Chapanis (1969) for a discussion of the merits and problems of the critical-incident technique.) I asked about customization strategies, specific triggers and barriers to customization, and the process by which users make customization decisions. I also incorporated new questions as they arose and interviewed several people a second time to incorporate additional questions.

The process of deciding to customize

What factors influence a user's decision to customize? When faced with a problem, users have a choice: either change their behavior or customize the software. The latter may take more time, but poses few risks. One manager observed that her staff "prepare personal cheat sheets, thus effectively customizing themselves rather than the software." Another user says that he only customizes when he has "the leisure time to do it. It won't take long, but if something goes wrong, I'll take a productivity hit." The alternative is to learn more about the system and risk spending an unknown amount of time making the change and possibly breaking something. Previous successes and failures affect the decisions here. One of the most common ways to reduce risk is to seek help from others. If someone else has already invested the time to make something work, then the chances that it will break are much smaller. *All* users in this study borrowed some or all of their customization files from other people in the organization (Mackay, 1990b).

Figure 1 summarizes users' decision process when learning a new software application and deciding whether or not to customize it. The first decision is whether or not to try the software. If the application promises a significant improvement over an existing application, but requires extra effort to learn, this may be a difficult choice. The user may decide to "play with it" as a way to determine how easy or hard this transition will be. If the user decides that it is worth the effort, he or she must then decide whether or not to customize the application. Such customizations are often made as another way to explore the software. If the user decides to make an initial set of customizations, they will influence the use of the software in the future.

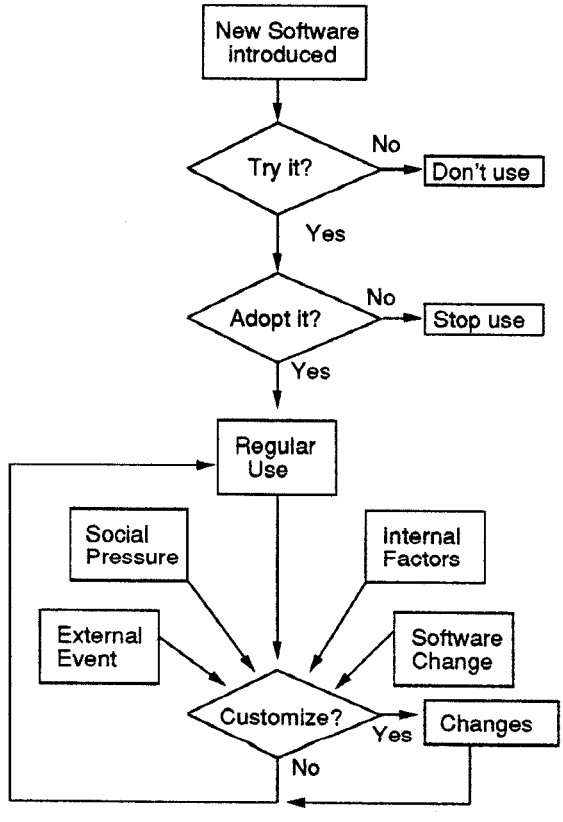

Figure 1: Decision process for customizing software

Users spend varying amounts of time and effort customizing, ranging from once every six months to every few days. However, users still spend most of their time simply "using" the software and concentrating on the tasks at hand. Usually, users need a reason to customize. The most common are external events that force them to stop and reflect on their behavior. Users identified the following kinds of reasons:

1. **External Events:** Job changes, office moves, personnel changes, going on trips (or experiencing an earthquake!) that force users to stop their normal work and think about how they organize their time and use of the software.

2. **Social Pressure:** Interruptions by colleagues who want to share or provide help or when the user sees something he or she would like to borrow.

3. **Software changes:** Breakdowns, upgrades, or the addition of a new program, may also force the user to spend time customizing. Interestingly, users often customize to maintain a stable interaction with the software, rather than to take advantage of new features. Mackay (1990a) found that all of the users who had customized their software at all included customizations that helped them maintain stable usage patterns, either by retrofitting the new software to be like the old or refusing to use the new software. 100% of those who had avoided customizing learned to use the new software.

4. **Internal Factors:** Boredom or sudden access to spare time (e.g. a three-day weekend or an evening in a hotel room while on a trip), or sudden insights about new ways of making the system do something, often in the context of trying something new or running across a previously unknown feature. Sometimes, users become fed up that something does not work properly and decide to fix it or discovers that he or she performs the same task repeatedly and devises a way to make the process happen automatically (e.g. creating aliases (shorthands) for moving from one place to another, setting options or arranging windows on the screen).

TRIGGERS AND BARRIERS TO CUSTOMIZATION

Table 3 summarizes the factors that users cite as both triggers and barriers in their decision-making process about when and how to customize software. The table is organized according to the four factors influence a user's decisions: the technology itself, the properties of the institution, external events, and individual factors. The factors are listed from most to least common in each category, preceded by the percentage of study participants who cited the factor. These data were compiled from the open-ended interviews or when users gave specific reasons for making or avoiding particular customizations. Participants were not given this list.

Participants identified 31 unique triggers and cited a total of 226, an average of 4.4 triggers per person. All but two participants cited at least one trigger and one person cited 11. Participants were most likely to customize when they discovered that they were doing something repeatedly and chose to automate the process or when the system changed and they modified the software to make it act as it did before 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 automating a common practice.) Observing something that another staff member does, either by walking by or when working with them, was the fourth most common. The next most common set of triggers include fixing something that no longer works, exploring the system when it is new, having someone set up the system when new, and creating a stable environment for people who need to switch from one to another (either from a machine at work to one at home or among machines at work).

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. By far the biggest barrier is lack of time, cited by almost two thirds of the participants (63%). "Lack of time" often means that the user is not willing to risk

spending an unknown amount of time on customization. One third cited lack of knowledge about how to make desired customizations (33%) as a barrier. Lack of interest and the general feeling that a particular problem isn't worth fixing are also cited.

Table 3: Factors cited as triggers and barriers to customization activity

Percent of Users	Triggers of Customization	Percent of Users	Barriers to Customization
Technology influences user			
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 info
Organization or other individuals influence user			
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 influence user			
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 influence user			
43%	Notice 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
51	Participants	51	Participants
96%	Percent of participants (49/51)	92%	Percent of participants (47/51)
4.4	Mean triggers cited per person)	2	Mean barriers cited per person

DISCUSSION

Although software manufacturers are increasingly likely to provide customization options, very little is understood about how users actually customize. We do know that users often resist using new software features. Mackay (1988a) found that 50% of full-time secretaries in a research study, selected because they had 1.5 or more years of experience with a text editor, had not learned basic editing functions such as "cut and paste" or "replace text string". Norman (1981) found that even expert Unix programmers used a small subset of the commands available to them, suggesting that lack of knowledge is not the only barrier to using the system.

The data in this study show that simply providing a set of customization features does not ensure that users will take advantage of them. Customizing involves a trade-off for users, who must choose between activities that accomplish work directly and activities that may increase future satisfaction or productivity, such as customization, learning new features, coordinating activities, or creating innovations. In each case, the user trades off a short-term investment for a longer-term potential benefit.

One can compare the decisions about 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 of using any of a number of software packages that are available.) Each software package 'depreciates' as other more effective packages become available or as new features are added to the existing package that must be learned. When do users stop their work and take the time to learn the features or customize to take advantage of them? 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' rather than optimize. They are busy and customizing takes time, so they only customize when they deem it worth the trouble and they understand how to make the desired changes. "Optimal" amounts of customization change with the user's work context. For example, a user may need to produce a report by 5:00 pm. She may decide that a customization designed to automate a procedure and save 20 minutes is less optimal than doing the same procedure manually, since the former poses a risk of not producing the report at all and she derives no benefit from turning the report in 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 most likely to be considered 'worth it'. Functions that become 'automatic', such as use of particular keys for particular functions, are very resistant to change and users like to retrofit the software back if the overall system changes. Unless the user is bored or just learning a new system, customizations that make the software environment aesthetically pleasing or more interesting are generally avoided.

Current software designs do not take into account that users are more likely to customize at certain times, that customizing is often a social process, and that certain kinds of events are likely to dramatically increase or decrease the probability of any individual user deciding to customize. They also don't recognize that users most often want to customize *patterns of behavior* rather than lists of features. Software designers should develop designs that allow users to learn about effect usage patterns and modify them as such. (Keyboard macros provide a limited version of this idea.) Users want information about their own use and that of other people with similar job responsibilities and attitudes from which they can base their customization decisions. Software designers should permit users to capture individual patterns of use and share them with others.

Managers should note the importance of external events on user's customization decisions. Artificially creating situations that allow users to stop and reflect upon their own interactions with the software will increase the probability that users will take the time to customize. Similarly, bringing users into contact with each other to explicitly share customization ideas will increase the applicability and usefulness of individual customization decisions. These are especially important because, otherwise, users are most likely to customize when they first learn the software, which is when they know too little to make effective decisions. Providing opportunities for customization later in the use cycle should increase the overall effectiveness of those customizations. Finally, managers should understand that users want to maintain existing, well-known patterns of behavior and will resist new features or software products that force them to change these patterns. Encouraging or providing customizations that enable users to maintain stable usage patterns will increase their ability to quickly use new software packages.

In summary, users follow common patterns when customizing their software. Some factors may trigger users to customize, while others may act as barriers and reduce the probability of customization. These factors include social pressure and other influences from the organization, external events, internal factors specific to an individual user and the characteristics of the technology itself. Understanding how thes factors influence users' decisions about when and how to customize should provide information to software designers, managers and users who create, introduce and use customizable software.

REFERENCES

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

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

Mackay, W.E. (1988). Tutoring, Information Databases and Iterative Design. In D. Jonassen (Ed.), *Instructional Designs for Microcomputer Courseware*. Hillsdale, New Jersey: Lawrence Ehrlbaum Associates.

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

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

MacLean, A., Carter, K., Lovstrand, L., and Moran, T. (April 1990). User-tailorable systems: Pressing the issues with buttons. *CHI '90 Conference on Human-Computer Interaction*. Seattle, Washington: ACM/SIGCHI.

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.

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

Trigg, R., Moran, T. and Halasz, F. (1987). Adaptability and Tailorability in NoteCards. *Proceedings of Interact '87*. Stuttgart, Germany.