USERS AND CUSTOMIZABLE SOFTWARE: A CO-ADAPTIVE PHENOMENON

by

WENDY E. MACKAY

B.A., Honors Psychology, University of California, San Diego
(1977)
M.A., Experimental Psychology, Northeastern University
(1979)

Submitted to the
ALFRED P. SLOAN SCHOOL OF MANAGEMENT
in partial fulfillment of the requirements for
the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1990

Signature of Author ______________________________
Alfred P. Sloan School of Management
May 21, 1990

Certified by ______________________________
Professor Wanda J. Orlikowski
Thesis Supervisor

Accepted by ______________________________
James B. Orlin
Chairman, Doctoral Program Commitee

USERS AND CUSTOMIZABLE SOFTWARE: A CO-ADAPTIVE PHENOMENON

by

WENDY E. MACKAY

Submitted to the
Alfred P. Sloan School of Management
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy.

Abstract

Co-adaptive phenomena are defined as those in which the environment affects human behavior and at the same time, human behavior affects the environment. Such phenomena pose theoretical and methodological challenges and are difficult to study in traditional ways. However, some aspects of the interaction between people and technology only make sense when such phenomena are taken into account.

In this dissertation, I postulate that the use of information technology is a co-adaptive phenomenon. I also argue that customizable software provides a particularly good testbed for studying co-adaptation because individual patterns of use are encoded and continue to influence user behavior over time. The possible customizations are constrained by the design of the software but may also be modified by users in unanticipated ways, as they appropriate the software for their own purposes. Because customization patterns are recorded in files that can be shared among users, these customizations may act to informally establish and perpetuate group norms of behavior. They also provide a mechanism by which individual behavior can influence global institutional properties and future implementations of the technology. The presence of these sharable artifacts makes it easier to study customization than related co-adaptive phenomena such as learning and user innovation. Because some mechanisms may be the same for all co-adaptive phenomena, findings about use of customizable software may also shed light on user's choices about when to learn new software and when to innovate.

Current research models do not provide useful ways of exploring co-adaptive phenomena, thus requiring new research models and methods. Research on technology and organizations commonly follows one of two theoretical models, each of which is inadequate to account for how users customize software. One treats technology as a static, independent variable, which influences the behavior of the people in the organization. The other treats the organization as the independent variable, in which decision-makers in an organization make strategic choices about technology and appropriate it for their own purposes. The structurational model proposed by Orlikowski (1989) takes both perspectives into account and incorporates an active role by individuals in the organization. This dissertation extends her analysis by examining the *co-adaptive* relationship between users and user-customizable software: users both adapt to the available technology and appropriate the technology, adapting it over time. These appropriations may take the form of user innovations which may change both the technology itself and the characteristics of the organization, such as who communicates with whom and how coordination of work processes is handled.

The theoretical model and evidence for co-adaptation is first illustrated with data from a pilot study of the Information Lens, a software application that allows users to customize the process of managing their electronic mail. I describe the development of the Information Lens and identify the interactions between the technology and individual users in the context of the organization. I also examine the individual patterns of use of Lens rules and trace patterns of sharing of rules among members of the organization. I then examine user customization of software in greater detail, in a study of Unix users at MIT's Project Athena. The data consist of interviews and records of customization files of 51 members of the Project Athena staff. The data is presented from the perspective of the structurational model, with a micro-level analysis of the customization decisions by individual users. The key findings include:

1. The specific identification of the interaction between users and customizable software as a co-adaptive phenomenon, supported by field data.
2. The theoretical linking of co-adaptive phenomena and the structurational model and evidence for a mechanism by which individual interactions with technology affect the organization.
3. The discovery of common patterns of customization:
    a. Users are most likely to customize when they first join an organization, which is when they know the least about the technology and their eventual use of it.
    b. Customization activities are often conducted as a way to explore a new software environment.
    c. Users attempt to incorporate their current work context into their customizations.
    d. Over time, most users make fewer and fewer customizations, regardless of level of technical expertise.
    e. Some external events, especially those that cause users to reflect upon their use of the software, increase the probability that users will customize.
    f. Users who customize like to maintain the same environment, even when the software changes. They will either retrofit the new software to be like the old or refuse to use it at all.
    g. The most common on-going customization occurs when the user becomes aware of a commonly-repeated pattern of behavior and encodes it as a customization.
4. Customization cannot be considered a primarily individual activity. The following patterns of sharing occurred:
    a. Users are most likely to borrow customization files when they first join the organization. These files are rarely evaluated for effectiveness and may have been created many years ago.
    b. A small group of highly technical individuals act as lead users of new technology. They are the first to explore new software and create a set of customization files that other people then borrow. However, the authors of these files receive little or no feedback as to the effectiveness or use of these files.

c. Less technical individuals take on the role of *translators* for other members in their groups. They interpret individual user's needs and create sets of customizations organized to meet those needs.

I conclude with a discussion of the theoretical implications, including support for and elaboration of the structurational model and the beginning of a theory of the use of customizable software. I propose changes in the software development process (to include observation of use in the field as an important input to future development), in software design (to include mechanisms that support reflection about use of the software and mechanisms for sharing of customizations), and for managers (to support periodic "maintenance" of skills and to support translators and help them provide more effective customizations for others in the organization).

Thesis Supervisor: Wanda J. Orlikowski
Assistant Professor of Management Science

Thesis Committee: Thomas J. Allen
Professor of Management of Technology

Thomas W. Malone
Professor of Management Science

Lucy Suchman
Research Scientist
Xerox Palo Alto Research Center

Marcie J. Tyre
Assistant Professor of Management
of Technological Innovation

Acknowledgements

I am deeply indebted to two people, who were instrumental in this dissertation. Professor Wanda Orlikowski gave generously of her time and I greatly enjoyed our many research discussions. She introduced me to a new body of literature and shared many insights. Professor Lotte Bailyn taught me new ways to look at data, took the time to read and carefully comment on my papers, and gave me encouragement and practical advice when I needed it most. Both provided a supportive intellectual environment and I cannot thank them enough.

I would also like to thank the members of my committee. Dr. Lucy Suchman was a source of inspiration and helped to change my ways of looking at the world. Professor Marcie Tyre gave challenging, insightful comments and helped me think more deeply about the theoretical issues in this research. Professor Tom Malone supported the work on the Lens study and encouraged me to think about the software implications of the research. Professor Tom Allen showed the importance of communication patterns in organizations and provided useful ideas about how to look at and present the data. Phyllis Reisner, Deborah Tatar and Randy Trigg gave me insightful comments on an early draft and I enjoyed interesting discussions about customization and the Lens study with Christina Allen, Danny Bobrow, Stu Card, Tom Finholt, Frank Halasz, Tom Moran, Jim Miller, Don Norman, Franklyn Turbak, Isaac Salzman, Hank Strub, and Ramana Rao.

I have learned a great deal from my fellow graduate students. I particularly want to thank Dietmar Harhoff, Abbie Griffin, Ben Whipple, Brad Hartfeld, Andy King, Jane Salk, Kevin Crowston, Dave Rosenblitt, Jin Tae Lee, Maya Bernstein, and especially Mark Ackerman, for challenging, on-going exchanges of ideas. Sharon Cayley deserves special mention for her practical advice, encouragement and occasional sanity checks.

I want to thank my first thesis advisor, Professor Murray Sidman, for teaching me the value of good writing and clear thinking. Professor Hal Abelson was responsible for my coming at MIT in the first place and provided a wonderful role model. Professors Andy Lippman, Glorianna Davenport and Chris Schmandt at the Media Lab kept me thinking about how all this relates to multi-media.

The staff at Project Athena patiently put up with all my questions and generously donated their time and I especially want to thank Beth Anderson, Paul Boutin, Don Davis, Eduard Guzovsky, Matt Hodges, and Brian Michon for many late night discussions. Professor Earll Murman and Ron Orcutt deserve special thanks for providing me with access to Athena.

I am grateful to Digital Equipment Corporation, for providing me with a scholarship and the opportunity to conduct this research. Many people at Digital were helpful over the years, in particular Jack McCredie, who first brought me to Athena and George Champine who continued to be supportive throughout. I also wish to thank Jim Gettys, Jim Miller, Win Treese, Vik Vyssotsky and Jan Walker from the Digital Cambridge Research Lab for their insights and discussions, Carl Klempner, for being a wonderful person and Branko Gerovac for becoming my technical sponsor and giving me useful advice. I'd like to thank Shirley Stahl, of the Graduate Engineering Education Program, for her understanding and help and Bob Glorioso and John Manzo for efforts far above and beyond the call of duty.

Sometimes, the best help is from people who are in a completely different field. I want to thank Krzysztof Blusztajyn, Barbara Slack, Ignacio Lopez, Mary Logue, Johann Sandmann, Ellen Garde, Misha Lakher, Marek Kloczewiak, Agnes Virga and the others, for Thursday night dinners with no mention of management. Also, my thanks to Bob McKie, Jill Bennett and the Border Cafe, and Michelle Fineblum, Debby Hindus, Ottavia Bassetti, Michael Granat, Bonnie Samet and Peter Brady for friendship and support. I also want to thank the members of the SIGCHI executive committee, especially my co-chair, Austin Henderson and Phyllis Reisner, Marilyn Mantei and Carrie Ehrlich, for helping me to stay in touch with the outside world and being so understanding during the final stages of the dissertation.

I want to thank my family, Mum, Dad, Heather and Trevor, for their love and support. Also, to George and Jean Brady, who encouraged me for many years. And finally, I wish to thank Steven Wertheim, for being there -- and actually making it fun.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Early computer software was designed to optimize the use of computer time, because computers were expensive and labor was relatively inexpensive. Computer users were either highly technical or expected to mold their activities to increase the efficiency of the computer. As computer costs have dropped and labor costs have increased, the emphasis has shifted. Software manufacturers have found it increasingly important to take the individual computer user's needs into account. They have directed some of the increased capabilities of the computer towards improving end-user productivity, designing new user interfaces (e.g. direct manipulation environments and interactive graphics), enhancing documentation and training, providing integrated instruction, making the software itself more flexible, and allowing users to personalize their interactions with the software by customizing it.

As software technology has changed, so has research about it and its use in organizations. Because the technology is a moving target, research strategies borrowed from the natural sciences that define the problem as predicting and controlling isolated natural phenomena have not met with much success. I am interested in the problem of how to study phenomena that *cannot* be isolated in this way, because they change as they are used. I have defined the class of such phenomena as *co-adaptive*, to capture the idea that people may both adapt to some aspect of the environment (e.g. computer technology) and at the same time adapt that aspect of the environment for their own purposes. Co-adaptive phenomena pose theoretical and methodological challenges and are difficult to study in traditional ways. Yet, some aspects of the interaction between people and technology only make sense when such phenomena are taken into account.

I claim that the use of information technology, particularly the use of customizable software, is a co-adaptive phenomenon. Customizable software provides a useful testbed for studying co-adaptation because individual patterns of use are encoded and continue to influence user behavior over time. The possible customizations are constrained by the design of the software but may also be modified by users in unanticipated ways, as they appropriate the software for their own purposes. Because customization patterns are recorded in files that can be shared among users, these customizations may act to informally establish and perpetuate group norms of behavior. They also provide a mechanism by which individual behavior can influence global properties of the organization and future implementations of the technology. The presence of these sharable artifacts makes it easier to study customization than related co-adaptive phenomena such as learning new software applications and creating user innovations. However, because the choices people make about adapting or adapting to the software may be the same for all co-adaptive phenomena, the findings about use of customizable software may also shed light on user's choices about when to learn new software and when to innovate.

This research has both theoretical and practical implications for the study of how people interact with technology. The theoretical questions include: Does use of customizable software constitute a co-adaptive phenomenon? If so, are there research models that provide a useful way to study these phenomena? Are people influenced by the software environment they use and and if so, how? Do people appropriate customizable software and if so, how? Does the presence of co-adaptive phenomena have implications for organizational theory?

The practical questions include: Which users customize software and under what conditions? Do individual customizations affect group norms of behavior and if so how? What are the implications for the design of customizable software and its use in organizations?

1.1 Co-adaptive Phenomena

I have introduced the term *co-adaptive phenomena* to describe the interaction between individual users and technology. "Co-adaptive" emphasizes the dual nature of the phenomenon: that people both adapt to technology (reacting to it) and adapt it to meet their needs (proactively changing it). Both processes occur over time and influence each other.

The choice of the term is influenced by recent changes in research in the natural sciences. The traditional scientific approach has been to isolate naturally-occurring phenomena and study them under controlled conditions. Several researchers (see Orlikowski and Baroudi (1989) for a discussion of these) have pointed out that the research methodologies designed to understand these phenomena are not appropriate for the study of technology, which is both the product of and is changed by human action.

Interestingly, aspects of the approaches being developed in the social sciences are beginning to be applied to the natural sciences. Evolution is an example. Evolution is a natural phenomenon in which plants and animals adapt to their environments. However, more recent research has identified a secondary phenomenon: plants and animals actively change the environment and these changes affect subsequent adaptations. The combination of these phenomena has been called *co-evolution* to stress the dual nature of evolution (Lovelock, 1979). Although the term co-evolution has been adopted by "new age" thinkers, it has also had an important effect on mainstream science.

A good example is the work being done by scientists at NCAR, the National Center for Atmospheric Research in Boulder, Colorado. They have concluded that in order to understand the earth's atmosphere, they must understand two related sets of phenomena: how plants and animals are affected by changes in the earth's atmosphere, and how their adaptations to those changes cause corresponding changes in the atmosphere. This research requires a global perspective and an interdisciplinary approach: "The role of the biosphere, the living material on the earth, is once again recognized as contributing to myriad chemical reactions, and new geochemical data are revitalizing research into the origins of the atmosphere...NCAR scientists, in collaboration with colleagues elsewhere, are now beginning the important work of linking our atmospheric models with equally complex and equally important biological, chemical and oceanographic models." (NCAR 1985 Annual Report, p.14, p.65)

An example of this perspective is the work by NCAR researchers Paul Crutzen, Paul Zimmerman and others, who were interested in the relationship between "biomass burning" in underdeveloped countries and the atmosphere:

> *The developing world resorts to clearing marginal farming land...by burning it off...This type of burning, which could be called nonindustrial pollution, may produce as much carbon monoxide, methane, and other gases as does the developed world...Crutzen led members of the NCAR Biosphere-Atmosphere Interactions Project in a tropical field program in Brazil called Project Brushfire in 1979 and 1980. Since 60% of the world's biomass resides in the tropics, this region has the potential for large emissions that can have global atmospheric effects...The measurements indicate that CO emissions from fires may make up as much as one-third of the global CO budget.*

Scientists were puzzled as to the source of high concentrations of methane in the atmosphere. They also observed the appearance of large numbers of termite mounds in the areas cleared by biomass burning.

> *During the biomass burning experiment, Zimmerman noticed that termite mounds were ubiquitous and especially common wherever fields had been cleared...Thinking that termites might prove to be a significant source of atmospheric methane...Zimmerman did field work in Guatemala and Africa to study the emissions of different species of termites. Laboratory studies at NCAR confirm that termites do release methane and can contribute significantly to the atmospheric methane budget.*

They eventually determined that the methane released from termite mounds, which has increased due to the burning of farmland, account for 24% of the earth's methane. Of course, these changes in the earth's atmosphere are having long-term effects on people and animals, who must adapt to the changes in the atmosphere or not survive.

Living organisms do not adapt to a static atmosphere, they adapt to a dynamic one and which they actively participate in changing. Their adaptations change the composition of the atmosphere, which in turn affects their subsequent adaptations. Just as we cannot think of the earth's atmosphere as a fixed variable that operates in predictable ways independently of living organisms on earth, we cannot look at technology as a fixed variable that operates in predictable ways independently of the people who use it. People adapt to new technologies, but they also appropriate and change them, which affects their future interactions with the technology. Thus, I claim that the interaction between users and technology in an organization is a co-adaptive phenomenon. Customizable software, which is designed so that users may adapt it for their own purposes, provides a useful example of a co-adaptive phenomenon.

1.2 What is customization?

Little is known about how users actually customize their software. Do end users typically customize their software, or is the practice limited to programmers who like to "hack" the system? Are there "optimal" levels of customization? If so, are the levels influenced by job requirements, individual expertise, social norms, time available, difficulty of customizing, or other factors? Are there optimal trade-offs between customization and performing productive work? If so, do how do users make these trade-offs and do they customize at optimal or even satisfactory levels? Are there barriers that impede the process 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? Do people customize along the dimensions indicated by the software design or do they customize in unanticipated ways, producing process or product innovations? Do customizations affect other members of the organization? If so, how? In order to address these questions, it is first important to agree on a definition of customizable software.

We all have an intuitive sense of the meaning "customization" because we have all customized in different ways. We arrange our offices and homes to suit our activities and aesthetics. Customization may involve adjusting a chair to the appropriate height, organizing files in a useful way or attaching notes to the telephone as reminders of frequently-called numbers. We may hang pictures, bring in plants, put up cartoons or post announcements of coming events. We use different strategies for managing our desks (Malone, 1983) and managing our electronic mail (Mackay, 1988c). When using software, we may prefer certain commands over others or set up procedures for accomplishing commonly-occurring tasks. We may use the customization features built into our software, to express our preferences about how it looks and feels.

While all of these can be considered forms of customization, for the purposes of this dissertation, I will restrict my observations to a subset of customizable software with the following characteristics:

1. Customizations affect an individual user's interaction with the software. Examples include: appearance (e.g. colors, backgrounds, fonts), level of difficulty (e.g. novice and advanced modes), default values (e.g. standard printers or values for variables) and process of interaction (e.g. keys associated with different functions or choices on menus).
2. Customizable software has a mechanism for allowing end users[1] to customize; users need not write code.
3. Customizations persist from session to session, until changed by the user.
4. Customizations are preserved in a form accessible to users, which may be shared. (Here, they are preserved as customization files.)

I distinguish customization activities from "normal use", which consists of the user's normal interactions with the software within the constraints defined by the software design. For example, changing the width of a column is a normal part of the use of a spreadsheet. Normal use may be simple or sophisticated and the range is influenced by the flexibility of the software. When multiple methods of accomplishing a task are available, users often prefer a particular method over the others, thus establishing a preferred pattern of use. Simply establishing standard patterns of use is

---

[1]End users are people who use software, rather than develop it. Unlike programmers, end users generally have fewer technical skills and tend to be interested in using the computer to solve problems that have little or no relationship to software development. Both end users and programmers are considered "users" of software.

David Liddle, Chairman of the Board, Metaphor Computer Systems and formerly of Xerox PARC describes the origin of the term "end user" (Liddle, 1990): "End users, in fact, are really interesting. 'End users' is a term I like, originally coined by IBM. 'End users' is a retronym. A retronym is a word that used to have a perfectly good meaning and now has to be retrofitted by a leading adjective, like an analog watch. You know, it used to be this thing with hands that went around, and then this miraculous thing, the digital watch appeared. And now all watches are digital. So if you happen to have one like this, you have to say that it's analog. Human-readable falls into this category. Anyway, we call them end users because 'Oh, you mean the end! Oh, that guy out there! I thought you meant the MIS guy that's more reasonable and rational. God! The end user?'"

not sufficient; the user must use a customization feature that continues to affect the software or the user's interaction with it in the future. Thus, repeatedly specifing that a column should be 10 characters wide is engaging in normal use. Changing the default value of the column width is customization.

I also distinguish between customization and programming. A programmer, using special skills and access, might, change the method by which it is possible to modify the width of a column. Programming changes are usually intended to affect the overall functionality of the program for all users, rather than to create a specialized form of the program for an individual user. (Note that this distinction becomes blurred for some software applications. For example, Apple's Hypercard includes a programming language called Hypertalk, which is easily accessible to users. Users may "customize" by changing a value or "program" by changing the structure or function of a particular application. Most software, and that examined in this dissertation, does not share this characteristic.)

I have chosen this subset of possible definitions of customization for several reasons. From a research standpoint, customization files are interesting because they persist; the users' patterns of use are encoded as artifacts that continue to affect an individual's daily patterns of behavior over time. These artifacts can be traced throughout an organization, in a manner similar to the way archeologists trace the spread of artifacts throughout a population. (Contrast this with other forms of communication, such as informal conversations, which may affect people at particular points in time but do not involve an artifact that exerts a continuing influence over time.) Systematic study of customizations and changes, compared across the individuals in an organization, should identify the factors that affect people's choices about customization and expose a communication network within the organization that has an on-going impact on behavior over time.

1.2.1 The design of customizable software

Software designers may provide customization capabilities through a variety of mechanisms. Users may modify "property sheets" (introduced with the Xerox Star) or choose from "menus" (made popular with the Apple MacIntosh). Users may create "initialization files" that can be edited with a text editor and executed when the software application is started. Users may also create and save "macros", which execute certain commands defined by the user. Some programs have "flags" that may be set which specify states or modes of use. The software may engage the user in a dialog, recording answers to a list of questions and setting options accordingly, or users may edit these directly. In each case, the software designer explicitly provides the "hooks" in the system to permit certain kinds of customizations by end users.

How many and what kinds of customization features should software manufacturers add? While we don't have good answers to this question, it's important to recognize that simply adding more customization features is not necessarily better. Most users do not want to solve "all possible problems"; instead, they want to solve a particular kind of problem or perform a particular type of task. The ability to customize the software in ways that have nothing to do with the task is irrelevant and wastes time. Thus, one of the jobs of the programmer is to limit the number of options available to the user, so that the user can spend his or her time on the solution to the problem, not the definition of it. The trick is to find a level of customizability that allows users to customize along the dimensions they find most useful, without burdening them with the problem of specifying every detail of how they want to work.

Customizable software should not be confused with flexible software. Software is considered "flexible" if it offers the user a variety of choices along one or more dimensions. For example, a spreadsheet is generally considered flexible because it allows users to set up a budget in a number of different ways. In contrast, a form-based budget program that restricts the user to typing data in a pre-defined way would be considered less flexible. However, either or both programs might be customizable. The spreadsheet user might be able to specify that columns are always 10 characters wide or that the background is always blue. The user of the less-flexible program might be able to specify the size of the font or the layout of entry boxes on the screen. In each case, the user specifies a preference once and the program continues to operate in that way until the user customizes it again.

1.2.2 The perceived value of customization

In many organizations, people have a choice as to when and how much to customize their software environments. How do people choose how much to customize? At one extreme, the user may choose to never customize anything. While this costs the user nothing, there is also no possibility for added benefits from customization, i.e. opportunity costs. At the other extreme, if the user spends all of his or her time customizing software, no work will be accomplished, and again, no benefits will accrue from customization.

For users who believe that some customization will be beneficial, it is often difficult to determine what an optimal or even desireable level of customization will be. The user must often balance an unknown cost (it may take a minute, it may take an hour) against a delayed benefit. If the customizations don't work, or worse, cause problems, then effort spent customizing may actually have a net cost. These costs and benefits are highly context dependent, and decisions are often made fleetingly, in the course of doing something else.

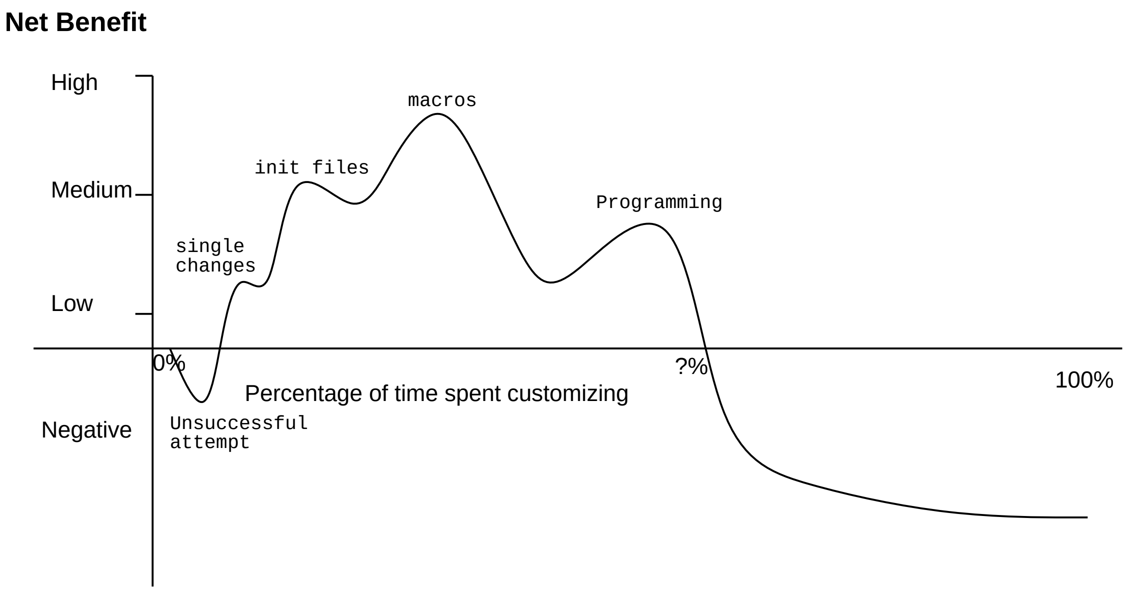


Figure 1-1: Net benefits vs. time spent customizing

Figure 1-1 shows an example of what the curve might look like for someone learning to customize a complex software package such as Emacs, a text editor. At the origin, no time is spent customizing and the user derives no benefits from customization. He or she may already be working at a satisfactory level and find no need for customization. Alternatively, the user may decide that customization would be benefitial, but has not yet tried to customize. An unsuccessful attempt takes time and may introduce problems, resulting in a perceived (and actual) net cost. A single, successful change produces a net benefit. Additional customizations may provide additional benefits. New customization strategies, such as initialization files or macros, may reduce the net benefit while they are being learned, but produce further peaks when successfully mastered. After a certain point, additional time spent customizing will no longer increase productivity (or satisfaction), and will begin to interfere with work. Learning to program (for the purposes of customization as opposed to accomplishing a job) appears to have a similar pattern of investment cost and potential for significantly increased benefit. At the extreme, users who spend 100% of their time customizing are unlikely to obtain any net benefit and will most likely be faced with a high net cost.

The above curve illustrates what an actual curve might look like, from the perspective of a particular user. It is important to emphasize these are perceived costs and benefits, very much embedded in the individual users' current context and work preferences. Thus, spending an hour to save three keystrokes for a function performed several times a day may be viewed as a significant benefit to one person and a waste of time to another. Costs and benefits cannot be evaluated easily, because users do not know in advance how long each type of customization will take, nor can they fully appreciate the benefits until the customization is complete. Thus the *perceived* costs and benefits will influence the user's choices about when to customize, rather than the actual costs and benefits. Users may weigh these perceived costs and benefits against a threshold, which may change over time, depending on the user's current situation. Each of the factors, costs, benefits and the threshold, are variable. Some are under the users control, others are not.

An informal cost/benefit analysis may help to explain why users make the choices they do and suggest ways to influence users' behavior. For example, if the perceived cost is higher than the real cost of a customization procedure, providing feedback about actual costs should increase the likelihood that the user will customize. Similarly, the decision threshold may change as the user's work environment changes. For example, someone rushing to meet a deadline will set the threshold for deciding to customize higher than someone who is bored. Providing incentives and tools to customize during slow work periods should increase the probability of customization. Describing the user's choice as a cost/benefit analysis has the disadvantage of implying that the costs and benefits may be measured independently of the current context of the user, which is not correct. However, this characterization does suggest way to help users make more effective decisions about customization.

1.3 Why study customization?

Users often 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. Some factors may act to trigger users to customize, while others may act as barriers and reduce the probability of customization. These factors may be similar to those that affect people's choices about when to learn new commands, read documentation, or create new methods of solving problems and understanding them may shed light on each of these areas.

Customization has the advantage of sharing many characteristics with learning and innovation, while being less open-ended and easier to study. Learning and innovation are difficult to define. Innovation has the additional problem of occurring rarely and is difficult to predict. Records of customizations are readily available, in the form of customization files, making it is possible to study both how users are influenced by different factors when they decide to customize and also how they influence each other through their customizations.

Although software manufacturers are increasingly likely to include customization features, very little is understood about how users customize or the effect of different customization designs on user behavior. We do know that users often resist using new software features. Mackay (1988a) found that 50% of the 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. Sondheimer (198) found that even programmers who had proposed ideas for new features for a new text editor resisted using them. Studying the factors that influence user's decisions about when and how to customize will help software designers design customization features and mechanisms and help users appropriate the technology in effective and productive ways.

Studying customization should also help us to understand some of the diversity in how people manage their work environments. For example, Mackay (1988d) found diverse patterns of electronic mail use, which reflected different job requirements, time management skills and strategies, communication needs, and level in the organizational hierarchy. These differences were reflected in the different ways the study participants wrote rules for customizing their handling of electronic mail.

Customization will become more important as more complex software environments[2] become widely available, particularly those that support multiple applications with different user interfaces. Until recently, the average number of applications for each office worker was 1.6 (Liddle, 1990), usually a text editor and a spreadsheet. This number is increasing rapidly. The software environment in this study allows users to run five or six applications simultaneously on the screen. These may be from different operating systems, run on different hardware, and exhibit very different assumptions about the user interface. Allowing users to customize the software by defining common forms of interaction across applications will make such systems significantly easier to use.

---

[2]The *software environment* includes the different software packages and the methods of access to files and other on-line data that people use to accomplish their work. Individual customizations, together with choices about which applications to use, make up the unique software environment for each individual. All of the individuals described in this research use high-performance multi-tasking workstations that are connected to each other via high performance networks and permit extensive sharing of data and programs.

Users do not always use the software in the ways specified by the software design. When software is used in productive but unanticipated ways, the result may be more appropriately termed "user innovation". When does user customization become user innovation? The difference may be simply one of scale, with "innovation" implying more substantial changes. The extremes are relatively easy to identify. Choosing a color from a menu or turning off the key click is customization, not innovation. On the other hand, applying a tool to a new problem, using it in a different way, or recombining it with other tools is user innovation, especially if the result significantly changes the behavior of one or more users in unanticipated ways.

Von Hippel (1988) has attacked the conventional belief that innovation is carried out solely by manufacturers, who see needs for new products and proceed to develop and market them. He identifies three potential sources of innovations: the manufacturers of a product, the users of the product, and the suppliers to the manufacturers. He predicts that the locus of innovation will occur where ever the greatest economic benefit accrues to the innovator. In the case of application software, the end user may have unique requirements unknown to the software developer. Thus, users would be expected to innovate if they receive relatively greater economic benefits than the manufacturers of a product. Many of these innovations are likely to be specific to the organization. Different members of the organization may benefit differently from different innovations. However, because they do not have the same levels of ability or access to resources, the benefits must be weighed in the context of the costs as well. This makes it more difficult to predict who is most likely to innovate within an organization.

The level of innovation is likely to be influenced by the design of the software and how it is interpreted by the user. The software primitives, and their inherent constraints, may provide users with more rather than fewer innovative ideas. For example, spreadsheets provide users with a powerful set of primitives: calculable rows and columns. While users are limited to working within this framework, it has proven useful for solving a wide range of problems. Users not only created new approaches to budgeting, the problem for which spreadsheets were originally designed, but they have also used spreadsheets for a variety of new applications. Some people use spreadsheets to manage columns of text. Others create data-entry forms and use them to enter data into databases or other software applications. Still others use spreadsheets to create simulations and educational games. One person even used a spreadsheet to create a working model of a circuit diagram, by using ones and zeros in appropriate cells and allowing the spreadsheet to calculate which circuits were operational.

Abelson and Sussman (1985) argue that a good program is not a solution to a particular problem, but rather a solution to a *class* of problems. They encourage programmers to build systems as levels of abstraction, with modular parts. The goal is to identify optimally-useful primitives at each level of abstraction. These primitives should then dictate the dimensions along which it is possible for users to use and customize the software. While these built-in constraints may have positive influences, as in the spreadsheet example, they may also influence users in less positive ways. Orlikowski (1988) observed that users of a new CASE system began to think about problems in a more constrained way and Zuboff (1988) described how office workers using a new dental claims system "stop remembering [the manual procedure] and rely on the system"(p. 133). Although the actual design affects what users can and cannot do, their perceptions affect what they try to do. These perceptions are influenced by users' actual interactions with the software, the context in which the software is used, their observations of how other people use the software, and their previous experiences and expectations about the software. Thus, there are a complex set of considerations to be made when designing software that supports customization and encourages innovation.

1.4 Research strategy for studying customization

This dissertation examines how users customize software and the effect this has on future adoption of new technologies. This work affects our understanding of how users incorporate software tools into their work lives and suggests implications for the design and process of introducing new software into organizations.

In order to understand how and why people customize their software environments, it is important to address these questions from an appropriate theoretical framework. Researchers disagree about which perspectives, assumptions and research methodologies are appropriate to addressing such questions. One stream of research views technology as a static element and measures changes in, for example, communication patterns in the organization (Sproull and Kiesler, 1986, Feldman, 1986, Gorry et al., 1988), organizational structure (Eveland and Bikson, 1988, Barley et al., 1988, Blau et al., 1976), and social control (Kling and Iacono, 1984). This approach could be applied to customization questions by examining how the introduction of user customizable software influences the behavior of members of the organization, perhaps in contrast to the use of non-customizable software.

Another stream of research views technology as a dependent variable and explores how decision-makers make strategic choices about the creation and appropriation of technology within an organization (Zuboff, 1988, Turkle, 1984, Noble, 1984). This approach could also be applied to user customization questions by examining the effect of management decisions about whether or not users were given access to customizable software and the corresponding organizational implications.

Each of these streams of research has been criticized for not taking the other approach into account and producing inconsistent and incompatible results (Kling and Iacono, 1984, Markus and Robey, 1988, Orlikowski, 1989). Orlikowski proposed an alternative framework that includes both the effects of a new technology on the organization and how the individuals in the organization affect and appropriate the technology. While the bulk of the analysis in this field is at an organizational level, Orlikowski's structurational model also accounts for the activities of individual human actors. It is not the goal of this dissertation to provide fundamentally new insights into these arguments in the theoretical literature. Instead, this research is presented within the model proposed by Orlikowski, with the goal of extending her analysis of how individuals interact with technology within their organizational settings.

An important goal of this dissertation is to demonstrate that individual use of technology is *co-adaptive*: that individuals both adapt their behavior in response to the technology and at the same time adapt the technology to meet their needs. Individuals are influenced by changing organizational factors, external events, and the evolution of the technology over time. However, individual users are not simply passive recipients of the technology, either as it is provided to them by management or by software manufacturers. They may use the technology in unanticipated ways, creating innovations that affect the technology itself (and perhaps the manufacturer of the technology). They may also share patterns of use, creating socially-defined norms of behavior that may ultimately affect the structure of the organization.

Chapter 2 describes the issues surrounding the theoretical models and proposes an extended version of the structurational model as the basis for this research. Chapter 3 illustrates the structurational model with examples from a study of the Information Lens. Chapter 4 describes the research method, site, and organizational context for this work.

Chapter 5 analyzes how individuals make decisions about how and when to customize and how customization helps them explore new software. Chapter 6 explores how groups of individuals share patterns of behavior and establish group norms by exchanging customizations. Chapter 7 discusses the implications of these results and relates them to the structurational model. Chapter 8 presents recommendations for the design of customizable software, discusses implications for organizational theory and user innovation and suggests directions for future research.

1.5 Glossary of terms

Normal use The use of application software as specified by the software design, with default values specified for customizable elements.(p.16)

Customization The act of modifying and saving individual usage preferences, along the customization dimensions specified in the software design. Customizations may be supplied by the user, by other members of the organization, or through sample customizations provided by the manufacturer. Customizations affect the user's future use of the software.(p.16)

Programming The act of modifying the code itself, not just parameters or values. Programming usually requires special skills and special access to the software, usually for the use of others in addition to the programmer. Programming may involve original development or modification of a software application and can change the dimensions along which other users may customize the software.(p.17)

Customizable Software Software that can be customized by an end user through mechanisms provided by the programmer. Customizations affect user behavior over time.(p.16)

Flexible Software Software that allows users a wide range of interaction choices within the software's basic design.(p.18)

User Innovation Novel or unanticipated customizations, software modifications or uses of software that provide a positive benefit.(p.21)

User Any person who interacts with a software application, including end users and programmers.(p.16)

End User Any person who interacts with a software application he or she did not write. End users generally have fewer technical skills than programmers and use the computer to solve non-programming programs. Programmers may also be considered end-users of software they did not write.(p. 16)

Programmer People who develop the software used by end users. Programmers may change roles, acting as end users of software they did not write and programmers or end users of software they did write.(p.16)

Application Software Software designed to help end users accomplish various tasks, including text editors, spreadsheets, and CASE tools. (p.17)

System Software	Software that provides support functions for users and software applications. Examples include managing the mouse and keyboard events and providing access to files and directories.
Software Environment	The set of software applications and methods of access to files and other on-line data, together with customizations, that people use to accomplish their work.(p.21)
Multi-tasking	The ability to interact with two or more software applications simultaneously on the same computer (usually a workstation).(p.21)

Chapter 2

Theoretical Framework

Managers in organizations create, adopt and adapt to new information technologies in an on-going effort to increase productivity and remain competitive. Although many researchers have studied this phenomenon, much of the work is contradictory and it is difficult to compare findings across studies. This is not surprising if we consider the diversity of academic disciplines (experimental psychology, sociology, computer science) and interdisciplinary specialties (organization theory and management science) involved. The purpose of this chapter is to describe and justify the theoretical model in which this research will be framed.

2.1 Competing research strategies

A number of researchers have attempted to articulate the problems associated with this diversity in research strategies (Kling and Iacono, 1984, Markus and Robey, 1988, Orlikowski, 1989). Kling (1980) concentrated on the *substance* of theory, such as concept definition and normative orientation. He identified and defined six theoretical perspectives in social analyses and empirical research on computing, which differ in their definitions of technology and social setting, theoretical constructs, evaluations of "good" technology and ideologies about the workplace. Markus and Robey (1988) analyzed the *structure* of various theories, based on the theorists' assumptions about the nature and direction of causal influence. They identified three forms of causal agency: the technological imperative (in which external forces cause change), the organizational imperative (in which people act purposefully to accomplish intended objectives), and the emergent perspective (in which change emerges from the interaction of people and events).

An example of the "technological imperative" is Sproull and Kiesler's (1986) work on the impact of electronic mail on communication within organizations. They argue that electronic mail does not simply speed up the exchange of information but leads to the exchange of new information as well. They found that decreasing social context cues had substantial deregulating effects on communication and that much of the information conveyed through electronic mail would not have been conveyed through another medium. Eveland and Bikson (1988) take a slightly different approach. They are interested in the ways in which electronic media influence the structure of work groups and their patterns of interaction. They conducted a field experiment in which they randomly assigned group members to either a computer-based or a traditional support group and gave each group the same task. They then examined the differences in communication patterns and volume of communication in general. In each case, the analysis is on the effect of the technology on the people who use it. They do not look for evidence of how the users appropriate the technology for their own purposes.

These examples are quite different from the "organizational imperative" research strategy. Zuboff (1988) examines how the choices of management can fundamentally affect the use of technology in an organization. She argues that technology can be designed to "automate" work, treating users as simply another component in a work process, or to "informate", empowering users to make decisions and operate autonomously. She feels that decision-makers in organizations should choose technology that empowers users, both for economic reasons, because users will be more productive and better able to handle unforeseen circumstances, and for humanitarian reasons. Yet Zuboff's analysis does not take into account the level to which individual users appropriate technology and make choices about use. In many computing environments, users are not simply passive recipients of information. They *react* to information, which changes their behavior and thus the behavior of the system. More importantly, they are sometimes *proactive*: they look for information from the system. They bring assumptions and experience and apply them to the technology in both anticipated and unanticipated ways. They may actively pursue information from the system, even if the system is not designed to provide it. Zuboff's point is that we can design software to encourage these activities, but it is important to note that some individuals will do it anyway, even without encouragement.

The third approach, the "emergent perspective" is illustrated by Barley's (1986) study of the introduction of CT scanners into two radiology departments. Barley found that the introduction of the same technology into two different organizations resulted in different organizational structures. His analysis includes a discussion of how the organizational structure is changed by the introduction of new technology. However, this analysis does not examine the effects of technology on individuals in any detail nor on how individuals affect the technology.

Suchman's (1987) work could also be framed as an emergent perspective, but in a somewhat different way. She criticizes the dominant assumptions of cognitive science and artificial intelligence, which assume that people "plan" their interactions with technology. She argues that it is essential to account for the "situatedness" of human social behavior attempting to understand how users interact with technology. She uses the term *situated action*, which "underscores the view that every course of action depends in essential ways upon its material and social circumstances. Rather than attempting to abstract action away from circumstances and represent it as a rational plan, the approach is to study how people use their circumstances to achieve intelligent action. Rather than build a theory of action out of a theory of plans, the aim is to investigate how people produce and find evidence for plans in the course of situated action." (Suchman, 1987, p.50) The implication for design is that artifacts by themselves do not determine their use, but rather the concrete circumstances of users' encounters with these artifacts determine use.

2.2 Structurational Model

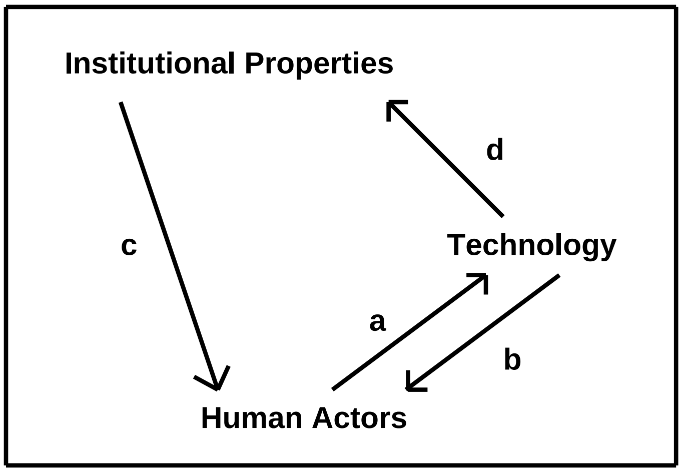
Orlikowski (1989) also identifies two streams of technology research, which she describes as the "technological imperative" model and the "strategic choice" model. Orlikowski proposes an alternative framework, after Giddens's (1984) concept of structuration, an "attempt to explicate how social practices are produced and reproduced across time and space...information technology can be seen to operate recursively: while on the one hand information technology is a product of social practices, it also facilitates further social practices (reproducing or even changing them)".(p.) The model is shown in Figure 2-1.

The interaction between human agents and technology is defined by two relationships: technology as a product of human action (*arrow a*) and technology as a medium of human action (*arrow b*). When viewed as a product of human action, technology is seen to be a human artifact, built within certain social and historical circumstances. It only has an influence because it is appropriated by people in the execution of certain tasks. When viewed as a medium of human action, it is seen to both facilitate and constrain the performance of tasks. The extent to which either dominates depends on the "intentions of the designers and implementors, the institutional context in which the technology is embedded, and the autonomy and power of the technology users."(p.)

In addition, human action is situated action (*arrow c*), shaped by the organizational context. Finally, peoples' uses of technology may operate directly on the institutional structure of an organization, (*arrow d*), either by reinforcing it or transforming it. According to Orlikowski, "These effects -- comprising the *unintended consequences of interaction with technology* -- are often not reflected on by users, who are generally unaware of their role in either reaffirming or disrupting an institutional status quo."(p.)

The structurational model makes a number of observations:

1. Technology is a human artifact, built within certain social and historical circumstances.
2. Technology only has an influence because it is appropriated by human action in executing certain tasks.



Key:

Arrow	Type of Influence	Nature of Influence
a	Technology as a Product of Human Action	Technology is an outcome of such human action as design & development, appropriation, and modification
b	Technology as a Medium for Human Action	Technology facilitates and constrains human action through provision of interpretive schemes, facilities, and norms
c	Unanticipated Conditions of Interaction with Technology	Institutional Properties influence humans in their interaction with technology, e.g. intentions, design standards, professional norms, state of the art in materials and knowledge, and available resources (time, money, skills)
d	Unintended Consequences of Interaction with Technology	Interaction with technology influences institutional properties of an organization, through reinforcing or transforming the systems of signification, domination, and legitimation

Figure 2-1: Structurational Model of technology (Orlikowski 1989)

3. Technology can condition but not determine social practices: human agency is required.

4. Technology is both facilitating and constraining, depending on organizational conditions.

5. Human action is situated action, shaped by organizational contexts.

6. Human action, through use of technology, acts upon the institutional structure of an organization, either reinforcing it or transforming it.

2.3 A Structurational analysis of a co-adaptive phenomenon

This research fits within the structurational framework advanced by Giddens and Orlikowski, with special attention paid to the interaction between human actors (users) and the technology (application software). Figure 2-2 shows the factors that affect a user, who must decide how to use the technology that is available and if, when and how much to customize it. Institutional properties include the organizational characteristics that dictate available resources and the expected standards of behavior within the organization. These may be modified by the expectations and practices of the local group, friends, or people who are located geographically nearby. External events also affect the user's decisions about customization. Some events are expected and planned for, such as an office change; others are unexpected, such as an earthquake. The constraints and opportunities provided by the technology itself also affect the user's choices about customization. Internal factors further affect the user's decision to customize, including the user's previous software experience, technical skills, expectations about job requirements and expectations about the technology. Examples of other internal factors include boredom, illness and the effects of previous decisions.

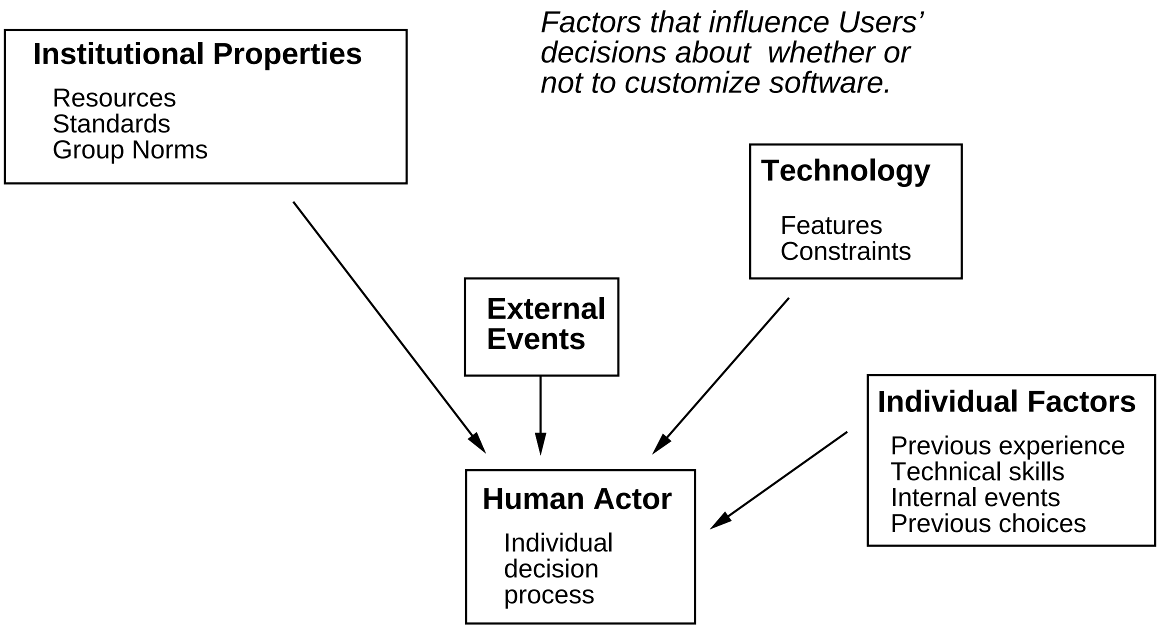


Figure 2-2: User-centered view of the Structurational Model: Inputs

Some of these factors may serve to trigger customization, such as an offer by a friend to help. Other factors may act as barriers to customization, such as the perception that a change would be too difficult. Users weigh the perceived costs and benefits against a threshold they define and decide whether or not to customize. Figure 2-3 shows the range of possible effects if a user does decide to customize the software. Users not only affect their future behavior, but may also affect the technology (through user innovation) or the behavior of others in the organization (through sharing of customizations or innovations). Some user innovations may be fed back to software manufacturers, which may then be redistributed back to the original organization via a software update.

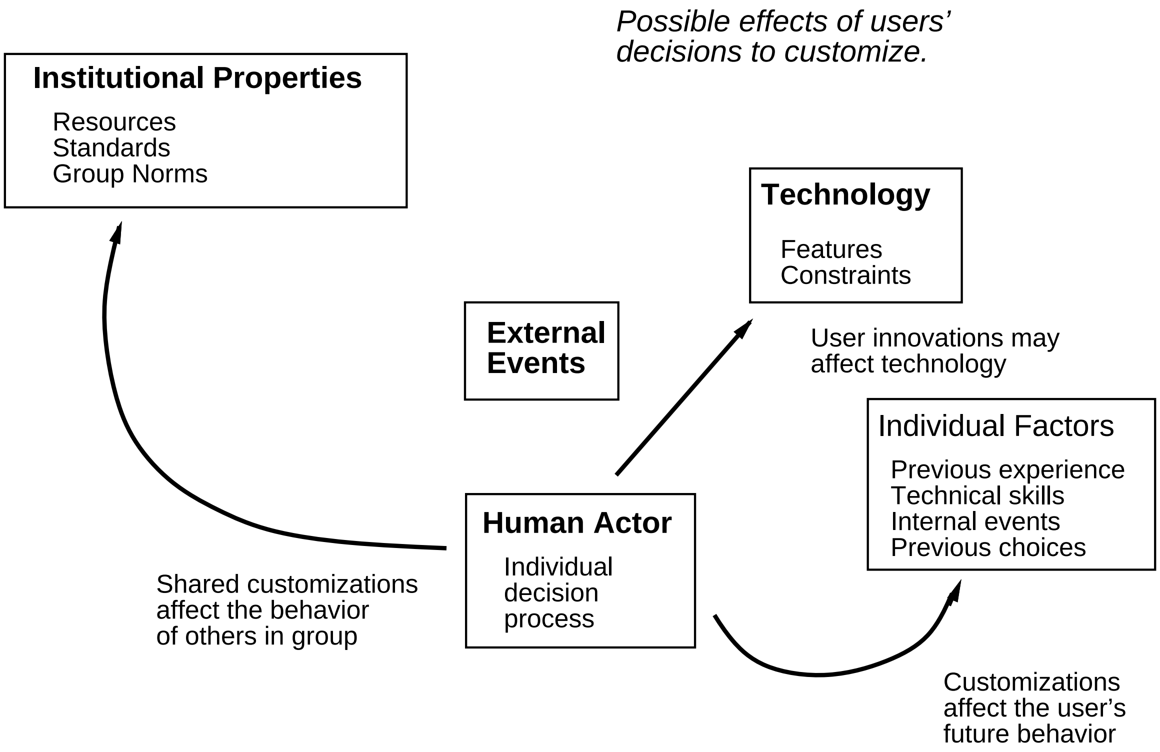


Figure 2-3: User-centered view of the Structurational Model: Effects

Figure 2-4 relates this user-centered perspective to Orlikowski's model. Heavy lines correspond to the arrows in Figure 2-1. Pairs of dotted lines indicate the user's decisions on future behavior and the effects of sharing customization files among members of an organization. The co-adaptive nature of the use of customizable software is especially apparent in these two sets of relationships. Users influence and are influenced by their use of the technology and their relationships with others in the organization.

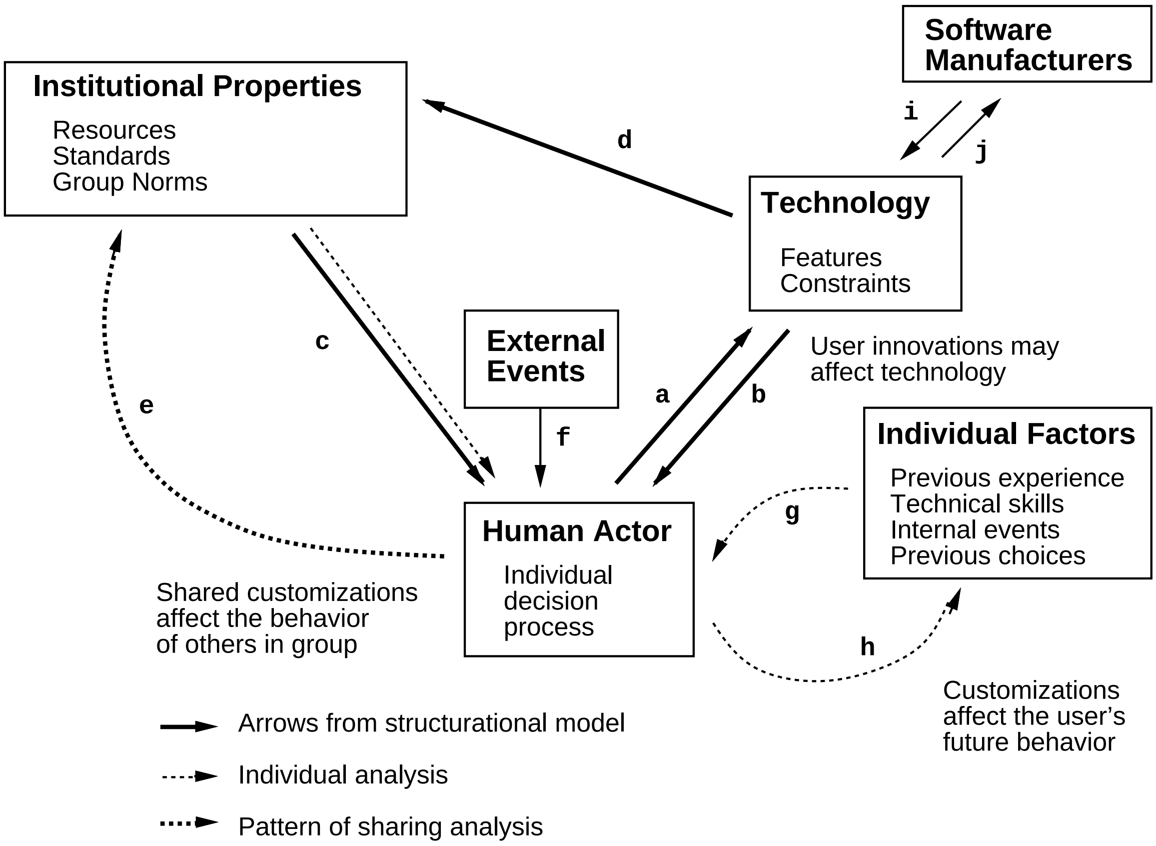


Figure 2-4: Relationship to Structurational Model

2.4 Research strategy

I am primarily interested in two types of questions: what are the factors that affect individual decisions about when and how to customize software and what are the patterns of sharing customization files within an organization. The assumption of co-adaptivity presents a problem when choosing a research methodology. Both questions assume that the interaction between people and software technology is situated in an organizational and individual context. I take as a fundamental assumption that software technology, as a product of human activity, is changed through its use over a period of time. These changes may involve changes in user activities that do not affect the operation of the code, customizations that do affect the code, but only in prescribed ways, and major changes through innovations or additional programming. Because of this perspective, a research method that treats changes in user behavior as the dependent variable and the software as an independent variable would be inappropriate.

Similarly, I take as a fundamental assumption that individual members of an organization are influenced both by the technology and by organizational factors. Individuals are influenced by how others use the system (either members of their group, friends, or people who are geographically nearby). These activities change the organizational context and group norms of behavior. A research method that treats the organization as the independent variable and the software technology as a dependent variable is also inappropriate, because it does not sufficiently account for the influence of the software design and individual actions.

In order to study these questions, it is important to examine the behavior of individuals within an organization over a period of time. (Isolated behavior within a laboratory setting would not provide useful answers to these questions.) It will be important to study individuals closely, to identify both common patterns of behavior and the diversity in their interactions with the software. It will also be important to study the ways in which they communicate with each other and share their understanding of the technology. In order to understand individual behavior, it will be necessary to examine the organizational context of the research site. It will also be important to understand the design of the technology and how it was intended to be used in order to understand the ways in which it is changed through use.

Because of the difficulty inherent in using a single research methodology, I plan to triangulate by collecting different types of data and comparing the results across data sets. I will record and examine the effect of organizational changes, external events, changes in the software and changes in the user's state on customization. I will also examine how a user's customizations affect the user's own future behavior, the behavior of others in the organization, and the software itself.

Chapter 3 illustrates the structurational model, enhanced by the co-adaptive perspective. I describe the changes in use and design of a prototype software system, the Information Lens, to explore how people were influenced by the technology and how they influenced and appropriated the technology for their own purposes.

Chapter 3

Illustration of the Structurational Model: The Information Lens

The purpose of this chapter is to illustrate the structurational model with an example drawn from the development and use of the Information Lens (Malone et al., 1987a), which allows users to customize the process of managing their mail. I begin with a description of the software and then describe the studies from which the data were gathered. I then discuss the organization in which it was introduced and trace the evolution of the software and its use over a period of one year, relating it to the structurational model. I analyze the data from an individual user's perspective, considering the triggers and barriers that affect choices about how much and when to customize. I also discuss the social aspects of customization, identifying who exchanges rules with whom and tracing how rules and other user innovations spread throughout the organization. I hope to show that a) users are influenced by the technology itself and other people's use of the technology, and b) users appropriate the technology and use it to meet individual and group needs.

3.1 What is the Information Lens?

The Information Lens is designed to help users filter and organize electronic mail and thus increase the effectiveness of their electronic communication. Lens accomplishes this by filtering *out* uninteresting messages (which allows users to process high-priority messages first and delete low-priority messages), and filtering *in* interesting messages, (which allows users to receive messages that would not normally be accessible). The net effect is to increase the effectiveness of mail handling by increasing the proportion of useful messages. Users can create their own sets of IF-THEN rules; incoming messages are then processed according to those rules. The rules usually search for text strings in the TO:, CC:, FROM:, and SUBJECT: fields and perform various operations, such as deleting the message or moving it to a specified mail folder.

The design of Lens involves three key ideas. The first is the concept of user-accessible *rules*. The rule editor was designed to look like an electronic mail message, to make it easy for non-technical users to identify items of interest and specify corresponding actions. The second key idea is the concept of semi-structured messages. Electronic mail messages have a set of structured fields in the header, such as TO: and FROM:, as well as an unstructured body of text. In addition to these fields, additional fields may be introduced to create *Message types* for specialized purposes. For example, members of an organization might use a message type called "phone message", which contains additional fields for the caller's name and phone number. A user might create one rule to look for a particular caller and display the message on the screen. Another rule might check to see if the caller was listed in the on-line phone directory, and if not, add the name and phone number.

The third key idea is the concept of the *anyone server*. Sometimes the sender of a message doesn't know everyone who might be interested in it. Rather than specifying a particular person or list of people, the sender can address the message to "anyone", which is the name of a central message server. People in the organization can also send sets of rules to the anyone server, to specify which messages they are interested in. The anyone server runs these rules on incoming messages and forwards messages accordingly.

The design and theory behind the Information Lens has been discussed extensively elsewhere (Malone et al., 1987a, Malone et al., 1987b), as has a newer system called Object Lens (Lai and Malone, 1988), which generalizes and extends the ideas from the Information Lens. Many software manufacturers have expressed an interest in the ideas and variations of Lens are being added to electronic mail products. I have reported about the use of electronic mail (Mackay 1988d, Mackay 1989a) and Lens (Mackay et.al. 1989a) elsewhere; the next section describes the research study from which these data were collected.

3.2 Research study

I conducted a research study in 1986-1988 to investigate how users manage their electronic mail and the effect of introducing the Information Lens on their mail handling processes. I wanted to observe how people actually use Lens "in the field", in the course of their daily work. This study was conducted in three phases, which correspond to three versions of the software. This was done for both logistical reasons and because the group of study participants changed with each new version of the software.

Pilot Phase: The first prototype of the Information Lens was developed at MIT in 1985-1986. The software was written in InterLisp-D and LOOPS and designed to work with the Lafite mail system. The prototype was used by a small group of software developers in the research lab at MIT. In Fall, 1986, I interviewed them about their use of Lens. This became the basis for the subsequent design of the interview and automatic data-collection procedures that were used at the research site. When the prototype of the Lens software was deemed sufficiently stable, an external research site was chosen to test the software.

Phase One: The prototype was brought to the research site in October, 1986 and I conducted preliminary interviews at that time. Due to a number of differences between the software at the two sites, the actual introduction of version one was delayed until January, 1987. Due to various bugs and other problems, all but one person abandoned it after several days. I interviewed these users about their experiences with the software shortly after this time.

Phase Two: Based on the experience with version one, the software was revised at MIT over a period of six months. In order to better understand and accommodate the needs of the users at the research site, the final set of changes were made by an MIT programmer who spent the summer there. I visited approximately one month prior to the planned introduction of version two to collect baseline data about use of electronic mail. Version two was introduced in late August, 1987. The entire lab was invited to a talk about Lens and a later "kick-off" meeting with a live demonstration of the software. Members of the lab were informed of these events and invited to participate in the study via electronic mail. 18 people volunteered to try Lens. I helped each person install the software and create a first rule. I conducted two sets of follow-up interviews at one-month and three-month intervals.

Phase Three: Version three was created by a local programmer who had actively assisted with the first two phases of the user study and had created a simple version for his own use. He then created a more substantial version, written in Common LISP and Common LOOPS, that could be used by others in the organization. This version was created to work with the new version of the Lafite mail system and the new version of the LISP system environment. Version three was smaller, easier to maintain, and incorporated the features most requested by users. It became the most popular version of the software and drew the most users, even though it lacked some of the features of the original Lens (including message types and the anyone server). Version three became generally available in January 1988 and support for version two was terminated. Since Lens users were only able to convert to version three when their workstations received the new LISP software upgrade, start dates for version three ranged over a period of several months. I conducted two sets of follow-up interviews three months and six months after version three was first introduced.

3.2.1 Research site

The site for this study was a large laboratory (approximately 60 people) in a research center for a multi-billion dollar international corporation. It was chosen because it met two key criteria: a) the hardware and software environment necessary to run the Lens prototype was widely available, and b) the members of the organization were experienced users of electronic mail who were currently facing the information management problems that the Information Lens was designed to address.

People at this site have been using electronic mail extensively for over a decade and a half. Essentially all employees at this site rely on electronic mail for both formal and informal communication; people receive an average of 30 to 35 electronic messages per day. Many of these messages come from a variety of distribution lists on both work and non-work topics. Most members of the lab, including managers and secretaries, have Xerox 1100 series workstations on their desks. These workstations are linked together in a distributed network that permits easy exchange of files and electronic mail messages.

Although this site is in some ways unlike many current office environments, it may be fairly typical of sites with lead users of advanced information technology. I believe that the experiences of people in this organization will resemble those of many other eventual users of advanced electronic communication systems.

Users interact with the software with a mouse and keyboard and multiple applications can be accessed simultaneously in different windows on the screen. Users usually choose applications via menus, although a command language is also available for access to applications that are used less commonly. Everyone uses the Lafite mail system, which has a graphical interface for composing, reading and browsing messages and mail folders. Messages are automatically marked with system-defined characteristics, such as *moved* or *deleted*, or with characteristics specified by the user. Lafite also has a concept of message templates, which users create when they need to repeatedly compose similar types of messages, e.g. bug reports and visitor logs. Users can create and name as many mail folders as they like. The Lens prototype was designed to enhance rather than replace this system. The goal was to allow users to chose any or none of the new Lens features, as desired. However, certain features, such as the use of the NEXT key and the ability to access one's personal message types were not available when using early version of Lens.

3.2.2 Participants

The lab is composed of research scientists, software engineers, managers, secretaries and other administrative personnel. Many members of the technical staff have Ph.D.'s in a variety of subjects, including Physics, Psychology, Computer Science, Anthropology, Mathematics and Sociology. The lab is divided into small groups which are responsible for one or more projects (some with outside funding). Each working group has a manager, secretary, one or more senior technical staff members and usually several junior-level technical staff. Many of the latter are students from local universities. The members of this laboratory are more highly educated and accustomed to more advanced computer technology than a representative sample of office workers. However, the participants in the study do include a range of job categories, educational backgrounds, and levels of computer experience.

Participation in the study was voluntary and different people participated in different phases; altogether, 26 people participated. The study was encouraged and partially supported by the head of the lab. The person who developed the local version of Lens was omitted from the analyses because many of his rules were created to test the system and could not be distinguished from his regular mail processing rules. In order to disguise the identities of the participants, their names and some of their personal characteristics have been changed.

Phase One: Six members of a small working group, including the manager, were asked to try version one of the software. All of these people were currently or formerly programmers.

Phase Two: Participation in phase two was solicited via an electronic mail announcement to the members of the lab. This ensured participation by active users of electronic mail. The message described the prototype of the Information Lens and asked for volunteers. 23 people, including two people from phase one, expressed an interest in trying version two. Five of these people ultimately chose not to use Lens, either because they were reluctant to try prototype software or because they did not trust Lens to handle their mail before they had seen it. Of the 18 who decided to try Lens, three were managers and 15 were full-time researchers. Of these, six were computer scientists and nine were trained in Physics, Psychology, Anthropology, or Sociology. The five who chose not to use Lens included three administrators, one computer scientist and one manager.

Phase Three: Members of the lab were told about the availability of version three via electronic mail. 26 people either converted to or began with this version of the software. Over half of these people had previous experience with an earlier version of Lens. Mackay et. al. (1989a) reported how a subset of this group, "experienced users" who had used version three for three to seven months, used rules. The 13 participants included a manager, a manager/secretary team[3], four computer programmers, one research scientist with formal training in computer science and six other research scientists (from a variety of physical and social sciences) without formal training in computer science. The people omitted from this analysis had either stopped using Lens or had not yet accumulated three months of experience with version three of Lens. The most common reason given for not continuing use of Lens was use of a workstation that did not support the new Lens software. Two people (one programmer and one manager) reported that they did not find Lens sufficiently useful.

[3]The secretary used Lens for both her personal mail and her manager’s mail.

3.2.3 Data

Two kinds of data were collected: participant interviews and automatically-collected records of use of the Lens software.

Interviews

I conducted three types of interviews. Preliminary interviews were designed to identify existing patterns of mail use, prior to using Lens. Installation interviews were designed to help the user install and begin using Lens, as well as provide a second look at existing mail use patterns. Follow-up interviews were conducted at varying intervals after the user had been using Lens and were designed to look at how users used Lens and its effect on their mail handling strategies.

All interviews were scheduled for one hour in the participant's office. Prior to each interview, I asked the participant to save one day of mail messages, deleting any that were confidential. In the preliminary interviews, I asked questions about the user's background, including programming experience and job responsibilities. In all interviews, I asked a series of questions about the person's use of electronic mail. I asked them to estimate the daily numbers of messages sent and received, the number of mail folders, the size of the inbox and the number of distribution list subscriptions. These answers were checked against the actual numbers for the current day and participants were asked if the day was typical. I also asked open-ended questions about major problems and successes with electronic mail. Participants used this an opportunity to describe their current communication patterns, successful mail management strategies, problems that needed to be addressed and whether or not particular Lens features would be expected to help.

In the installation interviews, I helped users install Lens and dealt with any system problems. During the installation time (approximately thirty minutes), I repeated the questions about electronic mail use from the preliminary interviews and asked additional open-ended questions. I gave each participant a five-page mini-manual, which explained the key Lens features on one page, followed by a three-page description of how someone might use it on a typical day, and a page with five sample rules. Those who requested it were also given a 45-page user manual (Mackay, 1986b). I helped each user chose a first rule, create it with the rule editor, create a test message, mail it, and check to see that the rule fired. Most people chose a variation of one of the rules suggested in the mini-manual. Users were responsible for creating additional rules by themselves.

In the follow-up interviews, I repeated the questions about mail use and then asked about their use of Lens, current choice of rules and what effects, if any, Lens had had on their general mail handling strategies. Interviews were conducted over a number of visits, shown in Table 3-1. The last three visits in phase two included a combination of preliminary, installation and follow-up interviews.

Table 3-1: Dates of interviews at research site

Phase	Start		End		Year	
1	10/13	Mon	10/14	Tues	1986	
1	1/27	Tues	1/29	Thur	1987	
2	7/20	Mon	7/24	Fri	1987	
2	8/20	Wed	8/26	Wed	1987	Kickoff
2	9/1	Tues	9/4	Fri	1987	
2	11/5	Thur	11/6	Fri	1987	
3	4/14	Thur	4/19	Tues	1988	
3	7/13	Wed	7/15	Fri	1988	

Phase One: I conducted one set of preliminary interviews and a set of follow-up interviews shortly after the participants tried to use version one. Because all but one person had stopped using it, I did not include many questions about use of Lens. I did, however, gather many suggestions for improving the software. One person continued to use Lens; he eventually developed the local version of Lens that was used in phase three of the study.

Phase Two: Most of the preliminary interviews were conducted one month prior to the introduction of version two. I also conducted several preliminary interviews for the people who volunteered to participate during the kickoff meeting. Installation interviews were scheduled immediately after the kickoff meeting. I also conducted a few installation interviews when I returned the following month for follow-up interviews. I conducted two sets of follow-up interviews at one-month and three-month intervals after the kickoff meeting.

Phase Three: I conducted one set of interviews three months after the conversion to version three. On this visit, I also asked participants to rate the lens features on a scale of 1-5. I conducted another set of interviews three months later. I noted the effects of converting from version two to version three of the software, which required users to recreate their rules. Most of the interviews in phase three were conducted as follow-up interviews, because most of the participants in phase three had been involved in phases one or two. Several people started using version three of Lens without joining the study; I did not interview these people.

Automatic data collection

The automatically-collected data were recorded for a period of over two years, beginning in mid-1987 and continuing through 1989. This analysis will concentrate on data collected in phase two and the first seven months of phase three. These data were recorded in LISP files (called "Lens profiles") and include the user's rules, Lens customization preferences, and the hierarchy of mail folders. Later versions were enhanced to record when each rule was created, changed, deleted or fired. A separate program translated the LISP file into a more easily-readable form. These data were collected weekly or every time the user changed any aspect of Lens. In addition, I also collected data about the user's distribution lists within the company. Members of the lab were placed on some of these distribution lists automatically and joined others voluntarily. (I did not have access to external distribution lists.)

3.3 Lens use in the context of the Structurational Model

3.3.1 What is Lens, really?

Just what is the Information Lens technology? Is it the software that is described in various research articles? Is it the original prototype? Is it the prototype that was most widely used? What about the Lens spin-offs, Object Lens (Lai and Malone, 1988) and Argus (Mackay et. al. 1989b), which generalize and extend different sets of concepts from the original Information Lens?

Clearly, any or all of these could be considered the Information Lens. Lens is a complex collection of ideas, expressed through several implementations and changed by both users and developers over a period of time. While it is convenient to talk about the Information Lens as if it were a unitary phenomenon, this practice has the unfortunate effect of implying that Lens is static. It encourages people to misunderstand the process by which the software itself is changed through design and use. Lens has been extremely dynamic; the perception of what it is continues to be shaped by the developers, users, researchers, and even the people who read about Lens technology.

I hope to demonstrate that Lens users did not simply react to a static technology; they actively changed it. They were influenced by how the software was presented to them, by their previous experiences with the existing mail system and other software, by their on-going experiences with Lens and by the experiences of others in the organization. For a few people, Lens fundamentally changed their mail processing strategies. Different users took advantage of different features; a few used features in ways that were not anticipated by the original software designers and were not supported in the functional specification. These activities influenced both the subsequent design of the software and how other members of the organization thought about and used Lens. Figure 3-1 illustrates relationship between Lens and the enhanced structurational model presented in Chapter 2. The following sections present data that illustrate each of these relationships. Together, they comprise the complex definition of Lens as it is used by this organization.

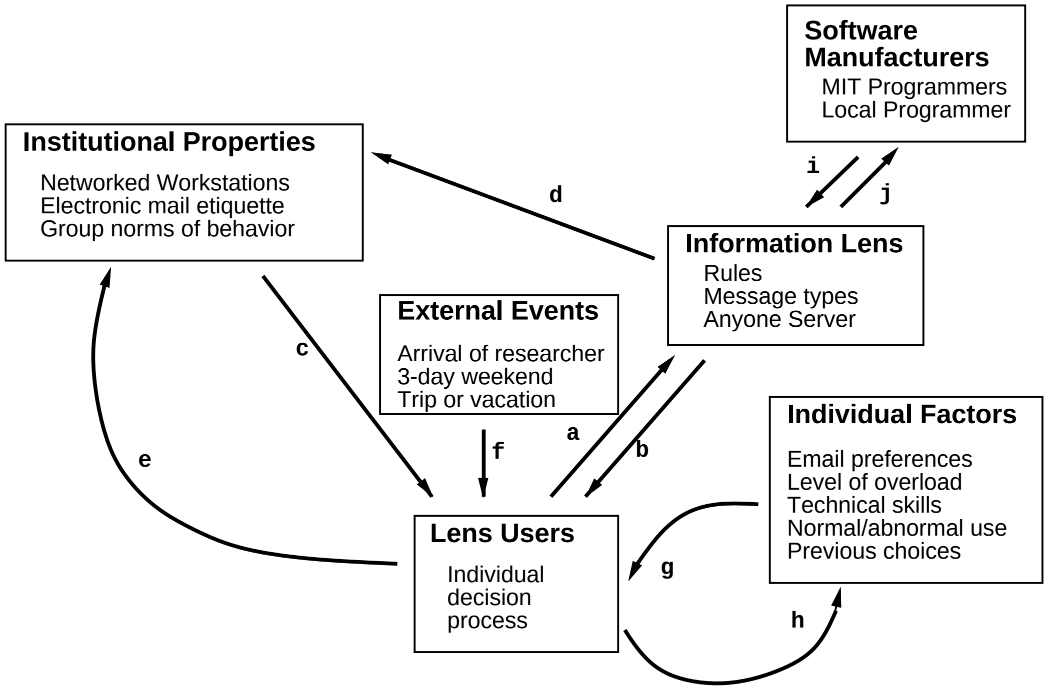


Figure 3-1: Structurational Model: Lens example

3.3.2 Arrow *a*: Users appropriate Lens

Arrow ***a*** in Figure 3-1 represents how individuals appropriated Lens for their own purposes. Mackay (1988b) demonstrated that members of this organization use the existing electronic mail system in diverse ways. In addition to the normal functions of a message system, they use mail for personal time management, for delegation and tracking of tasks within groups and for managing and sharing large quantities of time-sensitive technical information. Not surprisingly, these same users applied the Lens technology in diverse ways to support these different uses of electronic mail. A number of users simply followed the scenarios presented to them and used Lens as it had been described. Others made creative use of different features in the system, devising new uses for Lens. A few others actually created new software that was added to Lens, to provide additional functionality.

One example of a creative use of a Lens feature was by a manager, user ***A***, who wanted to use Lens rules to help her communicate with her secretary. She objected to using formally-defined message types, because they were slow, none of the existing types were appropriate for her purposes, and she could not create her own. So, she used a customization feature in a slightly different way to accomplish the same objective. The customization feature allowed users to add commonly-used alternatives to a menu, which became available during message composition. For example, the user might specify the usernames of the people he or she commonly sent messages to. The manager wanted to identify five categories (or "buckets"), which her secretary could then act on. Rather than creating five different message types, she added five code words to the menu for a single message type: *Please file*, *Remind me*, *Action*, *Please reply to*, and *fyi*. She would then read her mail, identify a message that she wanted to forward to her secretary for action, quickly append one of these code words to a field in the forwarded message, and send it. The secretary could then use her Lens rules to identify these code words and act accordingly. This is an example of how users adjusted the system to better support communication between two people who work together closely. It also demonstrates the user's desire to annotate incoming messages. The manager could not have created Lens rules that performed this categorization automatically, because the assignment was based on her knowledge and information in addition to the specific content of the message. Thus, the manager engaged in a two-step process of manual annotation followed by automatic processing.[4]

Another major innovation by users was also made possible through creative use of a customization feature. The original version of Lens was designed to run rules on messages prior to reading them. Version one used a concept of *rulesets* or groups of rules. Each user could specify a "local" ruleset, which fired automatically prior to reading new mail, and a "central" ruleset, which fired whenever the anyone server received new messages. Lens also had a debugging feature that allowed users to run the local ruleset on a particular folder. Many people had expressed an interest in the idea of

[4]Note: The manager later reported that this strategy was not as successful as she had anticipated: "I use the customized pop-up alternatives less than I expected -- the buckets are too big."

Lens, but did not like the idea that it would process their mail before they had had a chance to see it. One user discovered that he could use this feature to run rules *after* reading mail in the inbox. He changed his rules to sort his mail into folders, rather than to prioritize them for future action and ran this ruleset whenever he felt it was time to "clean up" his files. This was a novel idea. Another person commented: "I didn't think of reading mail and then sorting." When other users heard about this way of using Lens, they decided to join phase two of the study. Because of the success of this feature, version three was designed to explicitly support multiple rulesets. This was an example of a user-initiated change that affected the behavior of others in the organization (arrow ***e***, Section 3.3.6) as well as the future design of Lens (arrows ***i*** and ***j***, Section 3.3.9).

Table 3-2 shows what people did when they converted to version three and were able to create more than one ruleset. The "Pre-sort" strategy triggers rules automatically when new mail arrives. The "Post-sort" strategy allows users to manually run rules after they have been scanned or read. In versions 1 and 2, users appeared to belong to one of two camps: "prioritizers" who pre-sort and "archivers" who post-sort. However, once users had the option of using as many rulesets as they wished, 70% converted to using both kinds of rulesets. (One person created three kinds of rulesets). Of those who continued to use a single ruleset, one person stayed a prioritizer, one stayed an archiver, and one switched from archiving to prioritizing. This innovation thus spawned additional innovations, as users used different rulesets in different combinations to manage their mail.

One of the more subtle aspects of the introduction of multiple rulesets was the separation of the ruleset from the trigger for running it. Although automatic triggers (e.g. those that fire upon receipt of new mail) were clearly useful, manual triggers (i.e. those specified by the user) also had a special function. Many situations are easy for the user to identify (such as "I'm busy" or "I've just returned from vacation"), but difficult for the software to detect. In such situations, users find it easier to create rules without clearly articulating the conditions under which they will be run. When the time is right, the user simply runs the ruleset.

The importance of including the user when either annotating messages for future rule action or for identifying appropriate conditions for rule firing stems from a general problem: it is difficult for people to articulate appropriate behavior in sufficiently explicit terms for the computer. There are several reasons:

1. People rarely have access to data about their own behavior and their perceptions of their own behavior are often wrong.

Table 3-2: Conversion from single to multiple rulesets

User	Total V.2 Rulesets	Sorting Strategy	Total V.3 Rulesets	Sorting Strategy
A	1	Pre-sort	2	Both
B	1	Pre-sort	2	Both
C	1	Pre-sort	2	Both
D	1	Pre-sort	2	Both
E	1	Post-sort	2	Both
F	1	Post-sort	3	Both
G	1	Post-sort	2	Both
H	1	Post-sort	1	Pre-sort
I	1	Pre-sort	1	Pre-sort
J	1	Post-sort	1	Post-sort
	50%	Pre-sort	20%	Pre-sort
	50%	Post-sort	10%	Post-sort
			70%	Both

For example, Mackay (1988d) showed that none of the users were able to accurately estimate the quantity of distribution lists, and few were able to estimate the quantity of mail folders. Similarly, user's predictions of their future behavior, obtained during the preliminary interviews, often differed significantly from their subsequent use of Lens.

2. People have difficulty articulating their behavior and translating their desires into rules. When asked, they can make up rules, but these do not always correspond with their actual behavior. They usually refer to an "ideal" processing state and include a variety of misperceptions about technology and their own behavior.

   For example, one person commented about the *anyone* server: "I wouldn't know what to ask for." Another said: "I couldn't think of anything useful...It's a conceptual problem. I couldn't think of the different sorts [of messages]."

3. Some of the factors necessary for a particular decision are not accessible to the computer.

   For example, decisions about which messages to delete change according to how busy the user is at the time of the decision.

4. User's actions depend on the context in which the decision is being made.

   Suchman (1988) argues that the reason people find it difficult to articulate their plans of action is that "plans are resources for situated action, but do not in any strong sense determine its course...rationality anticipates action before the fact, and reconstructs it afterward" (p.52,53).

These factors help to explain why users were so pleased with the ability to create multiple, loosely-specified rulesets and run them manually, when the situation seems right. Users can now take their current work context into account without having to be more explicit about that context or adding extra work.

The use of multiple rulesets generated another insight about how users interact with mail. In the original design, rules were assumed to operate on messages based solely on content, length, or other attributes found in the message, independent of the user. However, consider what actually happens when a user receives a message. When it first arrives, the message is "new" and "unread" and potentially interesting. The level of interest may be affected by many factors, such as the user's current state (e.g. extremely busy versus bored) or the quantity of other messages (e.g. the only one versus one of a hundred). Most users scan the message header to determine whether the message should be deleted, acted upon immediately, or left for later. If the message is read and requires a reply, the length of time between the message's arrival and the current time becomes important. Thus, if the message arrived 20 minutes ago, the user is unlikely to be concerned. On the other hand, if it arrived two weeks ago, the user begins to feel guilty and feels that it is important to reply soon. Yet, this is a U-shaped curve: if the message is three months old, the reply probably doesn't matter anymore. (The guilt may fade more slowly.)

Other factors may influence the user's relationship to a message, such as changes in the user's job or in the organization. It is not possible or practical to pre-identify all of the possible factors that influence a user's changing relationship to a message. Yet the kinds of rules that are appropriate for handling it at any point in time may depend greatly on these factors. This suggests that users should be able to annotate their messages for future use. Just as user ***A*** customized messages he sent to his secretary, it would be useful to allow users to annotate messages as they read them, to facilitate subsequent processing by Lens rules. The Argus system at MIT's Project Athena (Mackay, 1989b) was designed to accommodate annotations (in the form of additional fields) created by senders, the mail system, and the recipients of the message. The key here is that the same message is treated very differently according to the user's context. A message is not a message is not a message: it is the relationship between the user and the message that is important. Users appropriate the technology in ways that take this changing relationship into account.

What is Lens, really? The definition has changed again. Users appropriated the technology, using it in novel ways. This influenced subsequent versions of Lens, which explicitly supported these appropriations. Users appropriated the new version, providing new insights into user's behavior and the usefulness of the software and influenced the software design once again.

3.3.3 Arrow *b*: Users are affected by Lens

Arrow ***b*** in Figure 3-1 represents how the introduction of Lens affected the behavior of individual users. The participants in the study were influenced in a number of ways. Perhaps the most subtle was the effect of hearing about the ideas behind Lens. Simply discussing the concept of automatic sorting of messages caused people to examine their own mail processing strategies, sometimes for the first time. They attempted to articulate these strategies, based on perceptions of their mail volume and their strategies for handling mail. The Lens demonstrations provided scenarios for solving mail problems, which led people to develop expectations about Lens and how it might help them. These perceptions and expectations formed the basis for their predictions about how they would use Lens.

Lens was initially presented as a sort of "automatic secretary" that could pre-sort mail into prioritized groups of messages. Some people liked the idea: "My goal is to read as little [mail] as possible." Others didn't: "I don't want automatic mail sorting because I don't trust a formula for sorting -- I'm afraid it would get sorted and I wouldn't look at it again." and "I could never trust a machine to read my mail for me." (Note that new managers often have the same reaction to giving up control over their paper mail to their new secretaries.) In the preliminary interviews, before trying Lens, individuals expressed strong and diverse opinions about different features. For example, some liked the idea of the anyone server: "The anyone server would be great." and "It would be very interesting if it could send me about ten messages a day." The second person hoped to use the anyone server instead of subscribing to different distribution lists. Others didn't like the idea: "It will only encourage more junk mail."

As mentioned in Section 3.3.2 (arrow ***a***), user's predictions often differed from their subsequent behavior. Perceptions of Lens evolved over time, influenced by experience, changes in mail handling strategies, and changes in other people's attitudes towards Lens. Most people did not evaluate their use systematically; only three people mentioned that they used their Lens profiles to evaluate the effectiveness of their rules. One user described how her views had changed through her own explorations: "Sorting seemed most useful at first, then you poke around and discover things." The presence of new options in version three affected users' perceptions of Lens and thus their decisions about how and when to use it. For example, the addition of multiple rulesets to version three caused eight out of ten people to change their strategies for processing their mail. One person, user ***H***, switched completely from one strategy to another.

User ***H*** had been reasonably satisfied with Lens in phase two. His rules had been designed to sort mail after he had read it: "Nothing happens automatically, I look, then sort." In January, he faced an impending mail crisis: he was planning to go away on a long trip and was also responsible for an upcoming scientific conference. At the same time, version two had been "retired" and he needed to recreate his rules in version three. Because he was in a hurry, he decided not to recreate all seven of his existing rules. Instead, he created a new rule that identified messages addressed directly to him (as opposed to messages addressed to a distribution list of which he was a member). He subsequently added one more rule, which identified messages about the conference and forwarded them to his assistant. When he returned from his trip, he found that these rules had separated out most of the important messages and left the less important messages. What began as an emergency measure to solve a crisis became his standard strategy for handling mail. He said these two rules "changed my life." He thus converted from a post-sort (archiving) strategy to a pre-sort (prioritizing) strategy. Although he intended to recreate an additional ruleset for sorting rules, he never got around to it: "I hate to stop long enough to get set up." These two rules significantly changed the volume of mail he processed. Instead of scanning the headers of all messages, he could now look at a significantly smaller subset of more important messages. Note that this changed how he partitioned the task of reading his mail. Instead of reading both important and unimportant messages all at once, he could put off viewing unimportant messages until he had time. The net result was that reducing the number of rules and running them at a different time increased the overall effectiveness of his mail handling strategy.

Many people reported changes in their use of Lens, their resulting behavior and in their perceptions of what they do. Some reported changes in their mail reading habits:

> *I don't read all my mail [anymore], I have a junk folder. I am pickier about what I read now.*

I read mail less often than I did before.

I used to read, then sort. Now, I sort directly.

Others began to attend more lab meetings:

I have attended more [lab meetings]. Without Lens, I probably would have deleted those messages.

I now go to [lab] meetings that I would have missed before.

I read all [research site] messages more carefully and attend more [lab meetings].

Some people developed new strategies for dealing with distribution lists. Three people reported that Lens made it easier to stay on distribution lists. One said:

I don't remove myself from distribution lists, I just auto delete the messages. Occasionally, I spot something in the deleted messages and take a look. I'm always disappointed and say to myself, 'Yes, I should have deleted it'. It's easier to change a rule than get off the d.l.[5]"

For others, Lens gave them feedback that enabled them to get *off* distribution lists:

I dropped off several d.l.'s.

[Lens] made me realize I was getting mail from useless d.l.'s.

Lens includes more than just the advertised features of rules and message types. Each implementation provides a number of additional features and characteristics that affect users in different ways. An example of a well-liked feature was a direct-manipulation interface for managing a hierarchy of mail folders, added to version two by the developer of version three. Figure 3-2 shows an example of a four-level hierarchy of mail folders displayed in this way. Users were able to transfer their folders to this format, and then rearrange folders to different positions in the hierarchy. As one person said: "I love the folder tree, I use it all the time...it encourages you to have different types of folders." Of the 23 people who tried it in different versions of the software, all but three reorganized their mail folders from a single level to a tree structure with two to four levels. This

---

[5]A "d.l." is a distribution list.

affected their use of folders: "I created more folders than I would have -- it comes in handy when I want to separate out [subcategories]. Otherwise they get too big...It makes everything easier to find." However, two people who tried it later decided they didn’t like it: "I generated a gorgeous tree of about 25 folders, but I only use 4-5...I don’t want my mail to be that complicated." Two others wished the folder tree took less space on the screen.

Figure 3-2: Version two: Folder Tree

Sometimes the effect of Lens changes as the user changes over time. For example, I interviewed a programmer who had recently joined the research lab (although not the company). At the preliminary interview, he was actively trying to identify which sources of technical information were important for his job. He had subscribed to 46 distribution lists, described himself as overwhelmed with mail and wanted Lens to help him deal with the problem. Six months later, he had stopped using Lens. He had decided that many of the distribution lists were "junk": "I got off a lot of d.l.’s...I’m down to a few general lists." This significantly reduced his volume of mail and he now felt in control of both of his job and the information he needed to perform it.

All of these examples demonstrate different ways in which people were affected by the changing Lens technology. Some people were affected simply by the discussions of Lens. Others used various Lens features and changed their strategies for handling mail. A few changed non-mail behavior as well, such as increasing attendance at lab meetings. The last example demonstrates how one user's needs for Lens were influenced by his changing relationship to his job. Lens initially helped him, by letting him identify unimportant distribution lists. He reacted by removing himself from the lists altogether, which eliminated the need for Lens. Thus, the technology can have a complex, and changing, set of effects on users.

3.3.4 Arrow *c*: Institutional properties influence Lens users

Arrow ***c*** in Figure 3-1 represents the effect of institutional properties on individual users. Institutional properties consist of the characteristics of the work and social environment in the lab. Examples include the computing environment, e.g. the use of LISP workstations connected via a high-speed network and the use of electronic mail as an essential medium for communication. Other properties include accepted norms of behavior, e.g. the etiquette in using electronic mail. The organization also supports several kinds of routine messages: A receptionist takes phone messages and sends them to the recipient via electronic mail. Members of the lab are expected to send a "visitor's log" message whenever they host a visitor. Seminar and lab meeting announcements are sent out via electronic mail and everyone in the lab is invited. A special program allows users to send hardware and software bug reports to the appropriate person, with all the necessary system information filled in automatically. These factors influenced the people in the organization and how they used Lens.

Social factors greatly influenced people's decisions whether or not to try each different version of Lens. Members of the first study group cited three kinds of reasons for using Lens:

1. *Technical reasons*: They found the Lens talk and ideas interesting.
2. *Practical reasons*: They wanted to solve existing mail problems.
3. *Social reasons*: They felt peer and management pressure to participate.

When the first group of users found the version one prototype slow and buggy, they made their objections known and others were reluctant to try version two as a result. However, the influence of several lab managers, the presence of several support people and revisions in the software encouraged other people to give version two a chance. Version three was the most well-accepted: it was written and supported by a local, highly-respected programmer, reflected many of their comments and suggestions, provided a simpler user interface based on the *Viewpoint*[6] standards, and several other influential members of the lab had tried it and liked it.

[6]*Viewpoint* is the name of the graphical user interface standard used at this site.

Social factors also influenced people's decisions to continue using the technology. Several people commented that they did not like early versions of Lens because their friends reacted poorly to extra empty fields: "[People] didn't like getting all these strange headers" and "the header has garbage that nobody wants to see." Another pointed out that the "length of the message affects readability and Lens lengthens messages." These people didn't like the fact that they had no control over how the final message looked and reacted better to version three because "it looks more like people generated [the message], not a machine."

Reactions to message types were mixed. Some people simply wanted to use the existing types: "I want the [Lafite message types], not the message type tree." Several people commented that they would be more likely to use the message types provided in Lens if other people did: "I avoided using message types...I don't see any point unless the whole community uses them." and "I would use [message types] more if more people did." One person felt that the message types that were provided were "too simplistic...like talking to Ann Landers." Another felt that his friends would think it silly if a request to borrow a book was labeled "request for action."

Users also resisted using software that was similar, but not identical, to their existing software: "The differences from Lafite are annoying...Lens redefines a lot of things." Three people insisted that a particular key, NEXT, be activated to operate as it did in Lafite. Part of version two was developed at the research site and users praised it for addressing more of their concerns: "The package improved immensely with [MIT programmer] here." Version three was the most similar to the existing mail system (Lafite) and was the most well-liked. However, several programmers chose to create rules in the LISP editor, instead of the rule editor, because they used the LISP editor on a regular basis and did not want to invest time in switching.

Some of the problems experienced with the early versions of Lens had to do with differences between the work environments at the development and user sites. On the surface, they were quite similar; the hardware and software platforms were the same. However, different patterns of use by individuals rendered certain characteristics of the Lens software unacceptable. For example, one manager objected to the length of time Lens took to update the Lens profile when he wanted to log out of his workstation. He had a noisy machine and a small office, so he liked to turn the workstation off when he held meetings there. Because Lens added an extra three to five minutes to the logout process, he decided to remove Lens from his system. Even though this method of saving profiles was later changed, he never resumed using Lens. The developers of Lens never experienced this problem because they tended to keep their workstations on and rarely waited for it to logout. The differences between the two work environments thus affected the perceptions of the software.

The lab has strong, though unwritten, rules of electronic mail etiquette. Many people commented unfavorably about breaches in etiquette and a few people cited etiquette violations as a key reason for using Lens. For example, the members of different groups or geographical locations are automatically placed on certain distribution lists. A common complaint was the indiscriminate use of these lists for trivial purposes, such as asking about lost keys in the parking lot. Another undesirable practice was the indiscriminate use of "cc:", copying a message to a large number of people for no apparent reason. This sometimes happened when a user replied to a message from a distribution list and the mail system automatically copied the reply to the distribution list.

Special interest distribution lists are similar to special-interest bulletin boards or news groups. People subscribe to the ones they find interesting and may forget they exist if the traffic is low. People describe these lists as having their own personalities and sets of expectations. Some generate and allow a great deal of "flaming", which involves heated exchanges over a previous message or hot topic. "Flaming" is expected and acceptable on some lists, but not others; people quickly learn that it's a bad idea to "flame" on the staff lists. Different people tolerate different amounts of flaming. Some tolerate a lot: "It's worth it to see 90% and delete in order to get 10% good stuff." Others do not: "So many people use email without etiquette. I got off a lot of d.l.'s and am down to a very few general lists...It's a shame." This person also commented that you have to watch carefully, because "good lists go bad." Another annoyance for people who read a distribution list is the behavior of newcomers, who ask questions and raise issues that have already been resolved. "We need a way for new people on a d.l. to go look at old questions and the etiquette for using the list."

One person realized that he could identify sequences of replies to a "flame". The mail system identifies replies to a previous message by placing "RE:" in the subject field and appending the rest of the subject of the original message. For example, if the header on the original message was:

To: *distribution list*
Subject: Tyrolian Ski Bindings

replies to the message would look like:

To: *distribution list*
Subject: RE: Tyrolian Ski Bindings

He decided that if the original message was not very interesting, the replies were not likely to be interesting either. So, his goal was to identify messages of the first type and delete all subsequent replies. He could have brought up a rule editor and created a unique rule for this purpose. However, he decided that it would be more convenient if he could simply press a button, as he was reading his mail, and append this new subject to an existing rule that deleted other boring messages. So, for example, if he already had a rule that deleted messages with the subject: "RE: Lehigh Safety shoes", when he pressed the button, "RE: Tyrolian Ski Bindings" would automatically be added to the list, as shown below:

If Subject: "RE: Lehigh Safety shoes" or "RE: Tyrolian Ski Bindings"
Then delete

Because he was a programmer, he was able to actually create this button and thus created the first "automatic" rule. Five people adopted this innovation and most found it very effective: "I use the boring rule to filter out flamers." The developer of version three added it as a regular feature of the software. Of course, these kinds of rules need not delete the message. One user suggested: "Why not have an 'exciting' rule, where you see an interesting message and automatically send it to the anyone server." Several others wanted to use the same idea to keep track of interesting or important discussions, saving them in a folder rather than deleting them. (This was later implemented by a student at MIT.) This is a nice example of a user innovation based on a reaction to violations of the norms of behavior in the organization. It also illustrates the importance of allowing users to evaluate and annotate messages as they read them, to facilitate future processing of that message (discussed in Section 3.3.2). This creates a partnership between the user and the technology, allowing users to add information that is inaccessible to the computer.

Another etiquette issue involves the use of network resources. "Autogetmail", introduced in phase two, automatically retrieves new mail messages for the user. Autogetmail was the highest-rated feature of Lens; one person even cited it as her reason for using Lens without rules. For some, it affected mail reading behavior: "My mail reading pattern changed -- now I read it when I want to, not when it arrives." Although they initially rated it highly, four programmers reported four months later that they had either stopped using it or used in under very restricted circumstances: "I don't use autogetmail anymore because of network issues...it keeps an open connection with the server...it's a matter of network etiquette." and "I don't run it continuously [anymore].". As technical people, they were aware of the impact of their actions on others in the organization and reacted by avoiding or restricting their use of autogetmail. However, less technical people, who didn't realize the impact of using this feature, continued to use it: "Autogetmail is really great...I would be offended if I had to get my own mail."

Institutional properties also affect the kinds of rules that people write. A good example is the use of deletion rules. Mackay et. al. (1989a) evaluated a group of rules from 13 experienced users and found that deletion rules comprise a small (20%) percentage of overall rules. 60% of rules that delete messages are qualified through specification of more than one field, whereas only 32% of rules that move messages are so qualified. Although a number of people feared that Lens users would write rules to automatically delete messages from them, this was never done. Personally-addressed messages were treated as significantly more important than broadcast messages. Only five of the 38 rules that deleted messages included the sender and all of these further qualified the deletion based on the content. These rules deleted messages from people who violated accepted standards of mail use, but were outside the range of peer pressure. One person was an outsider who had discovered how to reach the site and regularly sent messages about his new-age seminars. The other person taught aerobics classes several times a week and sent messages to the entire staff whenever it was time for another lesson. Although the authors of the rules that deleted such messages did not expect personally-addressed mail from either of these people, all were careful to include both the sender's name and another field that identified these messages. A personally-addressed message from either of these people would not have been deleted. These deletion rules were actually shared rules: one person created them and other people borrowed them.

People who write rules must deal with a trade-off when deleting messages. If a message is deleted, it will either be correctly deleted (correct rejection) or incorrectly deleted (false alarm). If a message is not deleted, it will either be correctly kept (hit) or incorrectly kept (miss). People base this trade-off on a number of factors. The most important appear to be the risk associated with not responding to a message and the person's desire to keep messages. The former is greatly affected by the properties of the organization and the consequences for any individual of not responding to mail messages. Some people can afford to incorrectly delete a message: "An accidental delete is not a problem." For example, if a manager does not respond to a message requesting something, the sender will try again, phone or stop by. Because managers often delegate tasks via mail, it is generally up to the recipient to respond. Managers also receive a higher percentage of messages for purely political reasons, such as messages copied to the manager to increase the perceived importance by the intended recipient. Managers and high-ranking staff can thus use stricter deletion rules and only worry about reducing the number of messages that are kept incorrectly.

In contrast, people in support positions, such as secretaries and software maintainers, often conduct much of their work via mail. One secretary commented that "most of the stress in my job comes from email." They cannot refuse most of the tasks delegated in this way and must process a higher percentage of their mail. They must also accept a higher percentage of incorrectly-kept messages in order to lower the possibility of incorrectly deleting messages that might be important.

The institutional properties described in this section, as well as others, affect the individuals in the organization in different ways. Factors such as the individual's role the organization, accepted standards of behavior and the characteristics of the computing environment, affect the choices they make about how and when to use Lens.

3.3.5 Arrow *d*: Lens affects institutional properties

Arrow ***d*** in Figure 3-1 represents the effect of the Information Lens technology on the institutional properties. Note that it is not the technology itself that affects the organization, but the technology as mediated by human action. Thus it is more accurate to say that arrow ***d*** represents how *users* appropriate and use the technology in ways that affect the organization. Because of the small numbers of users, it is difficult to identify any significant effects of Lens on this organization. Features that might be expected to have an effect, such as message types and the anyone server, were never used on a regular basis. However, many people expressed concern about the *potential* impact of these Lens features on the organization. For example, some people were afraid that the anyone server would increase the amount of junk mail: "Most people are too arrogant and most mail is junk...[the anyone server] would generate more." Others commented that it would be necessary for most members of the organization to agree to use message types before their use would provide benefit back to the individual users. The organization already had one example where the use of message types affected the organization. Members of the software support groups had reorganized their procedures for handling bugs when users started using standardized bug reports.

If arrow ***d*** had an opposite arrow, pointed in the other direction, it would refer to how properties of this organization, particularly its communication patterns and use of computing resources, affected the design of the Lens technology. The original designer of Lens was influenced by the specific characteristics of this organization when he developed the concept of Lens. Although a rule-based mail sorter was also being created at this site (Levy, 1986), users had to create rules in LISP and only programmers (and one researcher) used it. The idea of creating a rule editor that users interacted with as if they were composing a message allowed non-programmers the ability to sort their mail. The Lafite mail system at this site already supported a limited number of message types, including ones for reporting bugs and keeping track of visitors. Generalizing the concept of message types was a natural next step to increase the usefulness of the rules. This organization posted all "broadcast" messages through distribution lists, rather than electronic bulletin boards or conferencing systems. The anyone server provided a method by which people in the organization could obtain the benefits of bulletin boards, with greater control over topic, through their use of distribution lists.

The technology can influence the characteristics of the organization and, sometimes, these characteristics can affect the design and subsequent use of the new technology. Section 3.3.6 describes how users can affect these institutional properties both directly and through the use of the technology.

3.3.6 Arrow *e*: Lens use affects group norms of behavior

Arrow ***e*** in Figure 3-1 represents the effect of individual users on group norms of behavior and other institutional properties. Sometimes, a single person can affect the norms of behavior a large number of people. For example, the person who discovered the ability to run rules *after* reading mail (Section 3.3.2) affected the mail handling strategies of 17 people and increased the number of participants in the Lens study, including several who had previously refused to try it. Two thirds of the users (12/18) of version three developed a style that incorporated both strategies of using mail.

Sometimes, each individual plays different roles in the organization, which affects norms of behavior. For example, mail users play two different roles: senders and receivers of mail. Virtually everyone receives more mail than they send. Thus, from the perspective of the organization, it make sense to encourage senders to provide extra information, e.g. use message types, to make the recipient's mail processing task more efficient. However, senders rarely have incentives to do this. An exception is the bug report message type, which fills in information for the sender and thus saves time.

A few people send out most of the "institutional" messages. Thus, it makes sense to ask these people to format their messages in a standardized way. Because they repeat the task over and over, this not only saves them time and but also provides a more predictable message that is easier for the recipient to process. Good examples are the use of standardized phone messages by the receptionist and standardized seminar announcements by the secretary responsible for the seminar. Because most people in the organization *don't* send out these messages, it doesn't make sense to provide them in the sender's message type "tree". (However, it does make sense to make them available when people create rules for incoming messages.) People did not want to choose message types from a large number of infrequently-used alternatives: "I want to suppress most message types [and use] message types that are my own." Even phone messages, which may be sent occasionally by people other than receptionists, are unlikely to be created by ordinary users. However, several people were interested in creating message types specifically to help communication between pairs or among small groups of people. Two people wanted to create message types for their working groups: "I want to create my own message types for my group." and "It would be potentially useful for [group] messages." Three of the managers expressed an interest in using them for communication with their secretaries: "I want to be able to send [secretary] a 'to do' list with priorities, using code words." and the example described in Section 3.3.2 on page 44. The key here is to make sure that the costs for adding extra information are low and the incentives sufficiently high that individuals use the technology in ways that benefit the organization. If individuals cooperate, then they can increase the overall productivity of the organization and affect the expected norms of behavior.

Another way that Lens use affects the group norms of behavior stems from a common complaint: busy people object to learning about new technology: "My basic reason for not using Lens is that it's 'yet another widget'...I generally have a very conservative approach to learning new packages unless they're *very* useful... They're just that much more to learn and to screw up." These users rarely take the time to figure out how to adapt the technology to meet their current needs: "I didn't write any rules...[because I] couldn't think of anything useful." This resistance to spend much time learning something new lead people to ask for examples and learn from the experiences of others: "I'm busy and have a low tolerance for learning new things. I'd like a profile written by a Lens wizard." Asking for help from others has an important effect on the overall organization: a small number of people create patterns of using the software and others adopt those patterns. Choices about which pattern to use were influenced by the user's existing patterns of mail use and perceptions about Lens. Three basic patterns emerged:

1. *Prioritizing:* Rules that either move important messages out of the inbox into a "priority" folder or move unimportant messages out of the inbox, leaving the inbox as the priority folder.
2. *Archiving:* Rules that sort messages after they have been read, to facilitate subsequent retrieval of technical or other information: "I want to see it first, then archive it."
3. *Context-specific:* Rules that are run on specific occasions, e.g. returning from a trip: "When I'm back from a trip, I get tough. I'm vicious about messages... I wipe out seminars, etc."

When the ability to create multiple rulesets appeared, people talked to each other and decided to try each other's strategies. Two people discovered ways to copy all of another person's rules. In one case, the rules did not prove useful and he stopped using Lens. The exception was when he returned from a long trip and used the rules to successfully filter the mail that had accumulated. This encouraged him to start using version three of Lens. He created a completely new set of rules and then "borrowed" specific rules from the person he'd copied from before. The second person who copied an entire set of rules later edited them to be more useful. He recreated the rules somewhat differently when he converted to version three, after he had obtained some experience using the rules.

Although some individuals in the organization created their own unique sets of rules, many shared rules with each other. Borrowing rules has numerous advantages for individual users. They can reduce learning time and lower 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. Four people became the sources of these rules:

Researcher	Announced the Lens study, interviewed participants and provided a set of five example rules when they got started.
Developer	Created the local version of the software, based on own ideas and findings from user's experience with the earlier version of Lens.
"Regular user"	Non-technical person who liked and used the locally-developed version of Lens. Shared with other non-technical users.
"Lens wizard"	Highly technical, one of the first users of the first version of Lens. Immediately wrote almost 100 rules, provided feedback and ideas to the developer of version three.

As the researcher, I provided the most visible source of rules and rule examples. I did not provide any rules in phase one. However, in phase two, I selected five rules that illustrated the strategies that the participants in the previous study had found useful. Most of the participants in the study tried at least one of these rules.

The programmer who created version three of Lens gave advice about rules to approximately one third of the active Lens users, but did not provide direct copies for them to use. He introduced a number of innovations, which were incorporated into the design of the software and made available to everyone. One might have expected him to be the primary source of sharing rules, because of his knowledge of the software. However, he was more likely to provide advice and support than sample rules. He did not share his own rules, which were designed to test the system and were quite different from those of a normal user.

The "regular user" was a non-technical person who liked and used versions two and three of Lens. He adopted a prioritizing strategy from a colleague and then created his own, very stable, set of rules. Another manager in the group then borrowed his complete set of rules. This manager felt that they had similar mail needs and that the shared rules would provide a simple, useful model for her to get started with.

The "Lens wizard" was the earliest and most steadfast user of Lens. Considered one of the top technical members of the lab, he immediately began experimenting and created almost 100 rules. When asked why he experimented so much, he said: "Lens gets more valuable as more people use it...so somebody has to start. Throughout the project, he conferred with the developers and made numerous suggestions; he was also vocal with complaints. He created a number of innovations that were incorporated into the local version of the software.

The rule wizard created a number of complex, but useful rules that could be traced in other people's rulesets. A good example is the "Boring" rule described in Section 3.3.4, which was borrowed by five other people. Another example is a pair of rules that identify a particular author in a commonly-read jokes distribution list. The first rule identifies and saves jokes by this author; the second identifies all other jokes and deletes them. Three people borrowed this pair of rules exactly and another adopted a variation that identifies a different type of joke ('sniglets'). Two people borrowed his rules for deleting messages from two people described in Section 3.3.4 who violated the rules of distribution list etiquette. The function served by the rule wizard appears similar to that of the gatekeepers identified by Allen (1972). Gatekeepers are skilled individual contributors who actively seek technical information outside of the organization and translate it for the use of their colleagues. In this case, one person developed strategies for using rules, which he shared with his colleagues.

Rule sharing took another form as well. Two manager-secretary teams agreed on a standard form of communication and created rules to to facilitate processing messages to each other. In another case, members of a group agreed to customize their Lens rules to make their interactions with each other more efficient. These examples have a different character than the previous ones. They involve a social commitment to work together in a particular way rather than sharing primarily *individual* patterns of working with each other.

Although the number of users was quite small, and there were no facilities for exchanging rules, some users actively shared rules with each other. In each case, the rules were copied, used, evaluated and sometimes discarded. Most of the people who copied the "Lens wizard"'s rules were programmers with a range of 6 to 62 of their own rules who adopted one or two rules at a time.

However, people with similar jobs and interests also borrowed rules from each other. New users found it easier to try Lens because they could try rules that had already proven useful to others. Most people benefited from this sharing, taking advantage of each other's knowledge and ideas, 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 possibly errors) throughout the organization.

3.3.7 Arrow *f*: External events affect user's choices

Arrow ***f*** in Figure 3-1 represents the effect of external events on an individual's choices about when to use Lens. Although not included in the original structurational model, external events appear to play an important role in influencing when users decide to modify their rules. Although I do not have access to all possible relevant events, I have recorded several that appear to have influenced users: long weekends, organizational changes, trips, changes in the software, interviews by the researcher, and the conversion from version two to version three.

Two people reported that they liked to experiment with rules when they had extra time, usually on a holiday weekend. At least two changed their rules in response to a reorganization in the lab. At least three people reported that they modified their rules in response to a trip, because of the increased need for mail handling capabilities. In the example of user ***H***, described in Section 3.3.3, three external events occurred simultaneously, causing a change in his use of rules, which caused a significant change in his future mail handling behavior.

Table 3-1 shows the dates I visited the site and interviewed participants in the study. Six people modified their rules shortly after a follow-up interview. (Note: By definition, everyone modified rules during the installation interviews, so these interviews were not counted.) During one follow-up interview, a person commented: "I should redo my rules now that I've done it for a month." Although he had thought about the improvements he wanted to make, it was not until after the interview that he actually made the changes. Follow-up interviews often affected a user's perceptions of Lens, which could affect subsequent behavior. Not only did users learn about new features, but they sometimes were forced to question their use of Lens. For example, I asked people whether their rules had moved their messages correctly. Several people looked at me in surprise and consternation. It had never occurred to them that the rules might *not* work correctly and they began to worry about this new possibility.

Additions to the software, software problems that caused people to lose rules, and the conversion from one version to another also provided opportunities for people to re-evaluate their rules. Five people added the "boring" rule when it became available. Three people recreated rules differently after their rulesets were unintentionally deleted. Most people (87%) took advantage of the addition of the folder tree (Figure 3-2 in Section 3.3.3) to "clean up" their folders, sometimes adding new ones that made sense in the hierarchy and sometimes deleting ones that were no longer deemed useful. All ten people who converted from version two to three changed their total number of rules and nine of ten also changed their rule strategy. (The person who did not change simply recreated four of the six rules he had used before.) The effect of their conversions from version two to version three of Lens is shown in Table 3-3.

The conversion from version two to version three shows an abrupt change in most user's trend of increasing or maintaining the number of rules. Within phase two, four users increased the number of rules and five people tried it briefly and kept the same number throughout. Only one person decreased the number of rules. In phase three, eight people increased their number of rules, one person kept the number the same and one person reduced the number of rules. However, when these users shifted from version two to version three, eight people started over with a decreased number of rules. Only one person added new rules at this point.

One possible explanation for this abrupt change is that most users do not check to see whether their rules work and thus tend not to delete ineffective rules. However, when forced to recreate their rules, they also re-evaluate them. If they are deemed worthwhile, they are created again. If not, the rules are omitted. Thus the conversion from one version to another can serve an important role in improving the overall effectiveness of a user's rules.

Figures 3-3 through 3-7 show more detailed views of the patterns of rule changes for these ten users. The X axis indicates the months of the year in 1987 and 1988. The Y axis indicates the number of rules at the end of the month (the scale is indicated by the numbers to the left) and also the number of days that month that the user made changes (one notch on the Y axis equals one day). For most people, the conversion occurred in January. However, one person converted in December, and several others stopped using version two and waited for one to three months before converting to version three. Certain external events that appear to have affected rule changes are marked, including visits by the researcher, interviews by the researcher, long-weekends, and a staff reorganization.

User ***A*** spent several days after the installation interview trying different rules. After that, he only changed rules when there was an external stimulus: a follow-up interview with with researcher, a long weekend, the new release of the software and a staff reorganization. He increased his rules

Table 3-3: Patterns of rule changes from Version two to three

User	Version	# Rules at Start	# Rules at End	Lowest # - Highest #	Pattern of Rule Changes
A	Two	3	14	3-14	Steady increase in rules. Changed when researcher visited or on long weekends.
	Three	1	5	5-5	Follow-up interview: recreated 1. Added new rule every 2-3 days for 1 month. Added 1 after 3 months.
B	Two	14	13	12-19	Created 14. Added/deleted every 2-4 days for 10 days.
	Three	1	10	1-10	Recreated 1 rule. Added 4 more. Added every 1-2 weeks. Stable for 3 months.
C	Two	94	94	94-94	Copied rules from ***H***. Tried briefly, then stopped.
	Three	2	15	2-15	Follow-up interview: recreated 2. 3 days later, added 3. 2 months later, added 6. Added 3, at 1-month intervals.
D	Two	7	17	7-17	Highly variable, changed every 7-10 days.
	Three	17	9	2-17	Restarted with 17. 6 weeks later, restarted again with 2. Steady increase with changes every 5-15 days.
E	Two	7	7	7-7	Created 7 rules, then stopped.
	Three	12	14	12-14	Recreated 12 rules. Added 1 after 2 weeks and 1 more after 2 weeks. Stable at 14 for 8 months.
F	Two	9	11	9-11	Created 9, added 2 days later. Edited 3 weeks later.
	Three	25	39	17-39	Restarted with 25, deleted 8, added 1. Added rules once a month.
G	Two	75	98	75-98	Created 98 rules over several days. Then stopped.
	Three	23	52	23-52	Recreated 23 new rules. Added rules steadily over the next 10 months, occasionally deleting rules.
H	Two	7	7	7-7	Created 7 rules for sorting mail after reading.
	Three	1	3	1-3	Recreated 1 rule. Added 1 after 6 weeks and 1 more 6 weeks later. Stable for 6 months.
I	Two	6	6	5-6	Created 6 rules. Added 1, then deleted it.
	Three	4	7	4-7	Recreated 4. 1 added 4 months later. 2 added 2 months later. Stable at 7 for 2 months.
J	Two	23	23	23-23	Created 23 rules. Used for 2 months.
	Three	4	4	4-4	Recreated 4 rules. Accidently deleted 2 months later. Used autogetmail. Added 4 rules 2 months later.

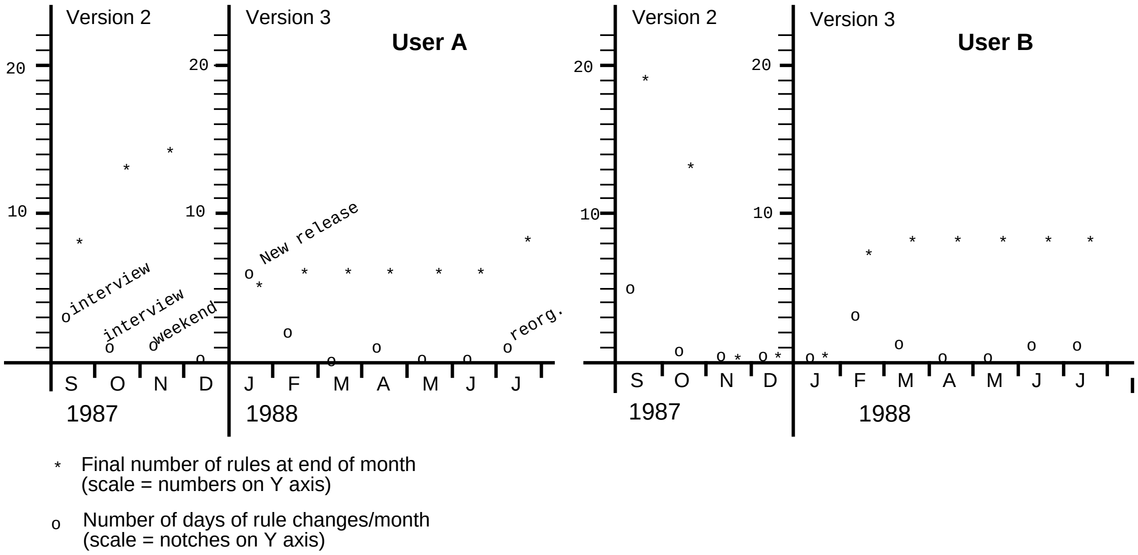


Figure 3-3: Conversion from version 2 to 3: Users A and B

from 3 to 14 during version two, and then recreated only 1 rule when he started over with version three. He slowly increased the number of rules and maintained the level at five or six for ten months. He is a very busy manager and carefully limits the amount of time he spends working with electronic mail.

User ***B*** created 12 rules and then experimented with adding and deleting rules over the next 10 days, with a total as high as 19 and ending with 13. He converted to version three in February when he created a single rule. Over the next four weeks, he steadily added new rules until he reached eight. He continued to use eight for four months, and then added two more. He tried a strategy of moving important messages into different folders and then discovered that he no longer read them. So he changed his strategy and now leaves important messages in his active folder.

User ***C*** began by copying a large number of rules from user ***H***, and then found them not very useful. He avoided using the system until April, when he had a follow-up interview. He decided to begin using Lens again and created his own rules, rather than copying them. He developed a steady pattern of rule changes, due partly to continued use of the boring rule, and was most likely to use Lens (and make changes) immediately after a trip. After a dramatic drop, from 94 rules to two rules, he steadily increased the number of rules over six months up to 15 rules. This person found that using Lens for handling messages after returning from a trip was most useful.

User ***D*** changed Lens rules often. During phase two, he changed them approximately every ten days, including when the researcher visited and on long weekends. The number of rules increased

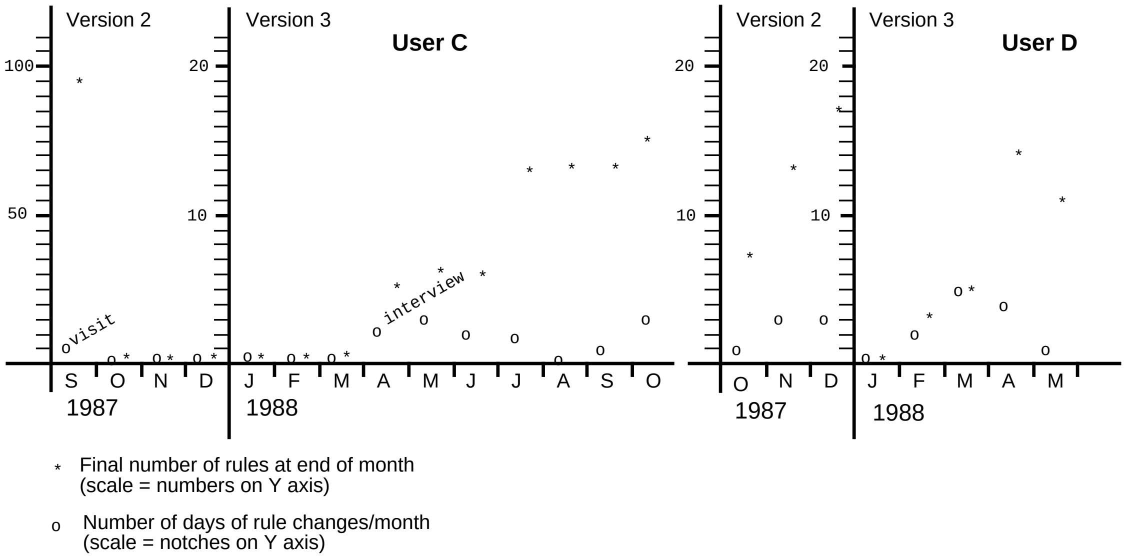


Figure 3-4: Conversion from version 2 to 3: Users C and D

from 7 to 17 in phase two. He re-created his rules in phase three and made changes every three-four days for the first few weeks and then every 10-14 days after that. He started with 2 and increased the number to 14. He often deleted rules and added new ones and checked the effectiveness of the rules by examining the Lens profile to see which rules had fired.

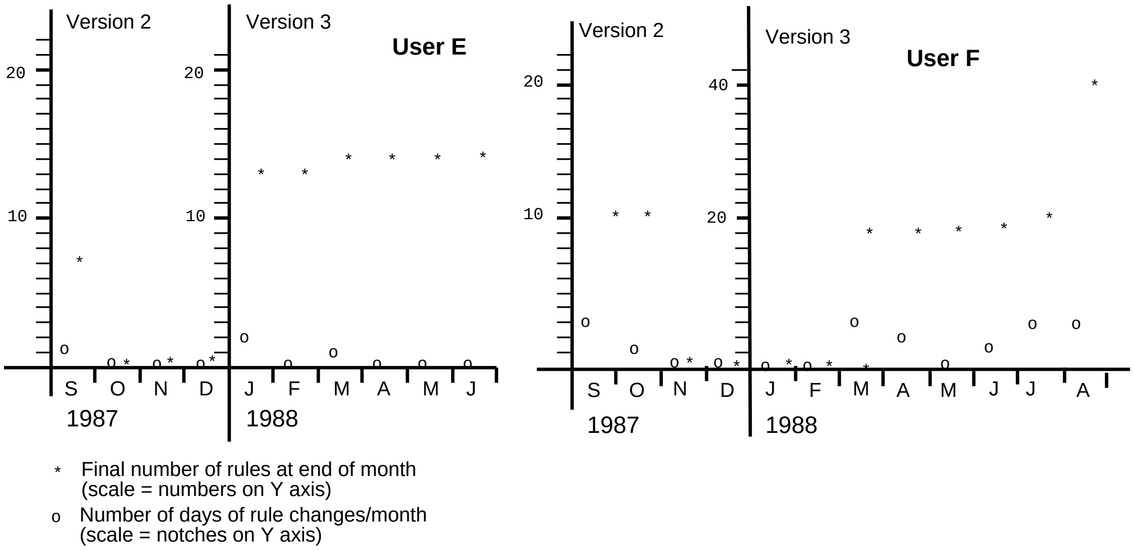


Figure 3-5: Conversion from version 2 to 3: Users E and F

User ***E*** created seven rules in phase two and then stopped because of system problems. He converted to version three in January and created 12 rules. He added one two weeks later and added another six weeks after that. His total number of rules remained stable at 14 for eight months.

User ***F*** created nine rules after the installation interview and increased the number to 11 by the end of the month. She had problems with her workstation and so did not beginning version three until March. She was unusual in that she recreated a larger number of rules at the beginning of version three than she had had at the end of version two. She created 25 rules, then deleted several and varied the total between 17 and 39 over a period of five months. She is interested in observing her use of mail and experiments with different techniques.

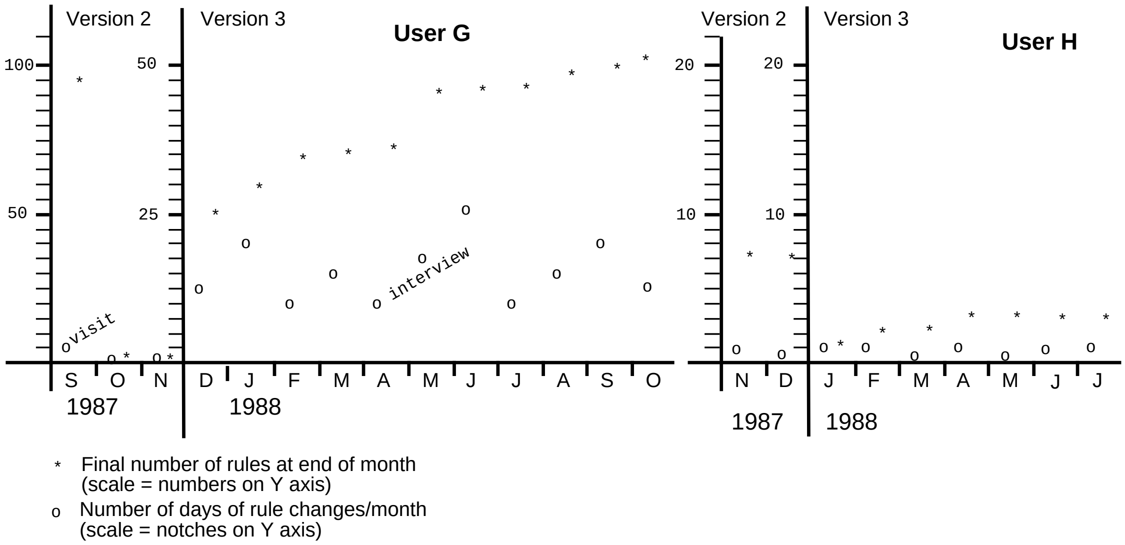


Figure 3-6: Conversion from version 2 to 3: Users G and H

User ***G*** created 75 rules, which he increased to 98 before he stopped using version two. He experimented with a variety of different kinds of rules and lent a complete version of his ruleset to user ***B***. He was the earliest user of version three and began using it in December. He started conservatively, with only 23 rules. Some but not all of the decrease can be explained by his use of the "to or cc" field, which replaced pairs of rules. He steadily increased the number of rules over the next ten months, adding one to five new rules every month. At the end of the study period, he had accumulated 52 rules. Because he has developed an interesting set of rules, other people ask him for examples and ideas.

User ***H*** created seven rules after the installation interview, primarily to sort mail after he had already read it. The conversion to version three coincided with going on a trip, so he created a single new rule that simply identified personally-addressed messages. Approximately one month later, he added another rule to identify messages about a conference he was organizing. After a follow-up interview two months later, he added a third rule that identified "boring" messages (described in Section 3.3.4). He maintained these three rules without further change for the next six months of the study.

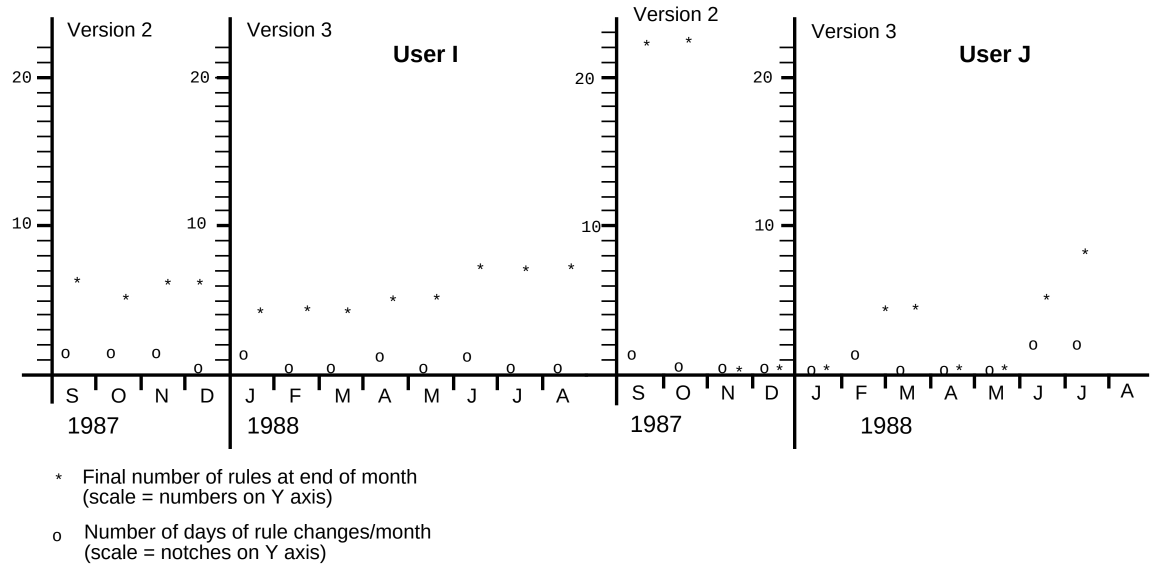


Figure 3-7: Conversion from version 2 to 3: Users I and J

User ***I*** created six rules after the installation interview. He recreated five of them in October, due to a system problem, and then increased it to six again approximately six weeks later. He was influenced by a strategy developed by user ***D*** and subsequently lent his rules to a manager, who started with version three. He recreated four of those rules when he converted to version three and added one rule after three months and two more after two months. Unlike user ***D*** and the manager, user ***I*** did not subsequently change his rule strategy.

User ***J*** created 23 rules and used them for two months before system problems caused her to stop. She recreated four rules and used them for two months before she was again forced to stop. She tried again two months later and recreated four new rules. The following month she added four more for a total of eight. For approximately two months, she used autogetmail without any rules.

As these examples show, external events may provide users with an opportunity to reflect upon their use of the technology and make changes accordingly. Once users must make changes anyway, the cost is sufficiently low that they make additional changes as well.

3.3.8 Arrows *g* and *h*: Individual factors and user's choices

Arrow ***g*** in Figure 3-1 represents the effect of past individual behavior, experiences and current individual state on the user's current decisions about how and when to use Lens. Arrow ***h*** in Figure 3-1 represents the effect of the user's decisions about Lens on future behavior and experiences. Individual factors are the set of predispositions, choices and behaviors that affect the individual user.

Mackay (1988c) identified at least three forms of appropriation of electronic mail by individuals, in addition to communication: time management, task allocation, and information management. It is important to identify the behavior patterns of individuals in order to discover that the technology has been appropriated in these ways. Individual analyses are particularly important when the behavior is highly variable or when individual patterns cancel each other out. They are also important where the behavior of interest is not the "average" behavior, but the unusual behavior.

Several extreme behavior patterns produced important insights. For example, one person found two rules to be very effective ("it changed my life"), while another wrote 98 rules but found them to be only "somewhat effective". By examining each individual, it was possible to gain an understanding of how the rules differentially affected their mail sorting behavior, and thus had differential effects on overall satisfaction. An example of the importance of individual differences is a comparison of two administrative staff members (not secretaries) whose jobs are largely comprised of sending and responding to tasks delegated via electronic mail. One person sends an average of 30 messages/day and receives 50-100/day (the highest averages of anyone I interviewed). He keeps 40 mail folders, subscribes to 56 distribution lists, and usually leaves with fewer than ten messages in his inbox at the end of the day: "I trim it all the time - it's like clearing my desk." (His desk was also completely clean.) He likes mail, has no trouble finding messages, and feels completely in control of mail. The other staff member is at the opposite extreme. He sends approximately 5 messages/day and receives about 23 messages per day. He subscribes to 68 distribution lists, keeps over 100 mail folders and keeps two active inboxes (totaling 1,000 and 1,350 at two different interviews), a number equaled by only one other person in the group. He occasionally creates a new folder to store the contents of the current inbox and then starts a new one. He says he likes mail, but he sometimes loses messages and admits to feeling completely overwhelmed most of the time.

Neither of these two staff members chose to use Lens. Perhaps not surprisingly, they cited completely different reasons. The first person felt that he already had a successful strategy for handling his mail and did not want to take the time to learn to use a tool he did not need. The other person felt so overwhelmed that he did not feel that he had the time to stop to learn how to use it, nor did he trust software that was clearly a prototype and might introduce new problems. He felt it was too risky: "I can’t afford to be a guinea pig."

The effect of individual factors may change over time, as in the example of the programmer who no longer needed Lens as he got his job and mail under control (Section 3.3.2). Several people discovered that they did not have the self-discipline to look into folders with important messages, if they had been sorted there automatically. One person described his experience: "I used to automatically move project-related messages into another folder, but then I didn’t check it. So now, if it’s important, it stays in "active"...it’s the calendar I look at every day. Most important messages stay in active. Otherwise, it’s ’out of sight, out of mind’." These changes in perception of their own behavior affected the user’s subsequent use of the software.

People spend varying amounts of time organizing their work, as opposed to doing it. Filing, creating piles, scheduling work, organizing papers, etc. do not accomplish work per se, but allow people to prioritize activities and get important things done. Sometimes organization activities serve to remind people about what they need to do next, make certain things easy to find, or simply place like projects together to reduce the cost of switching between tasks.

Another observation is that users who described themselves as "overloaded" were less likely to spend time using tools that helped them reduce overload. Not surprisingly, people who felt very much in control were also less likely to spend time organizing (they had either already invested the time and were reaping the benefits or their jobs or personalities were such that they were able to manage the flow of tasks/information.) The most interesting group were the people who were "on the edge" - basically handling things but at their limits. These people were able to try new things and were more innovative than their peers. They had a higher need than the people who felt in control, thus the benefits were higher. They were also more willing to pay the costs of organizing, because they weren’t completely overloaded. Panic induces paralysis.

Users who have very little incentive to modify won’t because the benefits aren’t worth the costs. Users who are overloaded have the incentive, but are not willing to risk the possibility of losing it all. Thus, we expect people who are in a state in which they are basically in control, but nearing the edge, to be the most likely to innovate/modify their working environments. Notice that this changes over time. When a user comes in contact with a new piece of software, the tendency is to move into "survival" mode first. The goal is to replicate the set of tasks that the user was able to accomplish before and try to get the job done. Over time, as that set of tasks becomes familiar, the user can do them more easily. At this point, the user moves from an "overload" situation to an "in control" situation, and is able to pick up more. Some users will stop here, others will begin to increase the demands of the job, taking on more responsibilities, etc.

3.3.9 Arrows *i* and *j*: Influences on future Lens development

Arrows ***i*** and ***j*** in Figure 3-1 represent the effect of changes in the technology itself on the developers of the technology and how those changes are reflected back to the users of the technology. The original design of Lens was influenced by and derived from a set of needs at the original site. The model of use was presented as the relationship between a secretary and a manager: the secretary's (and Lens's) job is to read and prioritize messages first, and then hand the prioritized set to the manager (or Lens user). As described in Section 3.3.2, some users chose to only use Lens *after* they had read their mail. This activity was initially enabled through a different use of a debugging feature, but subsequent versions of Lens allowed users to accomplish the same function by creating multiple rulesets triggered by different events.

The use of mulitple rulesets changed people's behavior yet again. It also influenced the software design. Because rules could now call other rulesets, it became important to identify the conditions under which a ruleset should stop processing a message. So, the developer of version 3 introduced the "stop" action. In order to avoid creating additional rules, he also allowed for multiple actions in the same rule. Thus a rule could now move a message to one or more folders, forward it, file it and mark it as "replied to". The implementation made certain assumptions about how people work; some made things easier for users and some made them harder. For example, the differences between the use of the NEXT key in Lens and Lafite caused problems for users. One of the easiest ways to see the changes in the technology is to look at the changes in the rule editor. Figure 3-8 shows an example of a rule created with the first rule editor. The rule has been given the name "Information Lens" and looks for messages that contain the word "Lens" in the subject field. When a message matches, Lens moves it to the folder called "Lens".

Figure 3-9 shows how the rule editor looked in phase three. The user interface was designed to meet *Viewpoint* user interface standards. Notice the addition of menu buttons and of multiple rule actions. In this example, the user has specified that messages addressed to or copied to the **Want Ads** distribution list, which also contain the subject **Honda**, are to be moved to the **Car** mail folder. Whenever a message meets these criteria, the rule fires and the message is moved to the **Car**

Figure 3-8: Original Lens Rule Editor: Version one

folder. The field labeled TO OR CC: was a feature added in response to many requests by users. In previous versions, this example would have required two separate rules: one addressed *to* Want Ads and one *copied* to Want Ads.

The software developers in this study were closer to their users than most. Changes and ideas from users were fed back in interesting ways. One of the effects of this was that the users taught the developers new ideas about what Lens is, extending it in a number of different ways. Through observation of actual use, the developers examined these needs and developed a new, richer understanding of the software. Lens inspired two independent development efforts at MIT, Argus and Object Lens.

Figure 3-1, which shows the Lens technology as a single box, does not adequately illustrate the dynamic nature of the software over time. If time is incorporated into the diagram, Lens should be a series of boxes which appear differently as users change their use of the software and as the software itself changes over time. Lens is not a unitary phenomenon; it changes both through use and through on-going development.

Figure 3-9: Revised Lens Rule Editor: Version three

3.4 Discussion

Table 3-4 summarizes the issues discussed for each of the arrows in the structurational model, shown in Figure 3-1. This study has demonstrated some of the diverse ways in which individuals react to and appropriate an information technology, the Information Lens. Each person uses some features and disregards others; two people may use completely non-overlapping sets of features. Previous experience with other software, differences in job requirements and mail handling patterns, and differences in personal characteristics all affect the user's expectations about what Lens is and can do. Perceptions about Lens and perceptions about their own needs affect their subsequent interactions with the software and the effectiveness of those interactions affect their on-going perceptions of Lens and Lens use. By observing how users actually used Lens, the developers were able to re-examine their own perceptions of the software and enhance subsequent versions of Lens to take more of the user's needs into account.

The findings from the Lens study suggest that users are likely to be influenced by a variety of factors when deciding to customize their software environments. Because they are busy, most people resist spending much time customizing. Certain kinds of external events appear to increase the probability of customization activities. These include events that affect the technology, e.g. software upgrades or system problems, events that change the user's rule needs, e.g. a trip or an organizational change, and events that cause users to reflect about their use of the software, e.g. talks, visits by the researcher, and viewing of the Lens profile. In addition, some users share their customizations with their colleagues. This sharing affects group norms of behavior and serves to spread both innovations and errors within the organization.

This study illustrates the importance of considering not only the consequences of individual interactions with technology, but also the effects of sharing of technology among members of an organization. Decisions made by individuals may affect the use of the technology by the organization, especially if those decisions are embodied in exchangeable files that continue to have an impact on user behavior over time.

The sharing of rules is a subset of a larger set of issues, including how users learn to use new technology and the ways they innovate and appropriate the technology for their own purposes. As users learn a new technology, they are influenced both by its design and the established patterns of use by other members of the organization. At the same time, individuals innovate. By sharing these with other members of the organization, they act to affect the structure of the organization. Software does not remain static when it is introduced into an organization. People in the organization evolve their individual patterns of use, share them with each other, react to external

Table 3-4: Summary of arrows in Structurational Model

Arrow	*Definition and Examples*
a	**Users appropriate Lens**
	Customization features are used as message types. Rule strategy is changed through use of multiple rulesets.
b	**Users are affected by Lens**
	Lens ideas affect thoughts about mail processing. Lens rules change mail reading and other behavior. Other features (autogetmail, folder tree) affect behavior. Lens changes user's use of d.l.'s, which eliminates need for Lens.
c	**Institutional Properties Influence Lens Users**
	Perceptions of Lens by peers affects Lens use. Reactions by recipients of messages affects Lens use. Existing message types, keys, user interface affects Lens use. Etiquette rules and desire to control abuses affects Lens use.
d	**Lens affects institutional properties**
	Insuffient use here to demonstrate this arrow. Institutional properties here affected design of Lens: rule sorting, message types, anyone server.
e	**Lens use affects group norms of behavior**
	Some people share rules and affect other's behavior. Small groups use it to communicate in predictable ways. Individuals can retaliate against mail abusers.
f	**External events affect user's choices**
	Technology changes: software failure, software upgrade. Reflective changes: interviews, talks, Lens profile. External changes: reorganization, trip, earthquake. Social changes: other people share rules.
g & *h*	**Individual factors and user's choices**
	Experience with LISP affects choice of rule editor. Feelings of control over mail affects choice of use. Changes in job over time affect continuation of use. Willingness/desire to experiment affects use. Past behavior and experiences affect future use.
i & *j*	**Influences on future Lens development**
	Design influenced by perceived needs. User innovations affected future design features: multiple rulesets, boring rule, to:/cc: field, Argus.

changes, both technical and non-technical, and sometimes proactively modify the system to produce significant innovations.

Chapter 4

Research Method

4.1 Research method

This study is designed to examine the customization behavior of an organization, the staff at MIT's Project Athena, in the context of the Structurational Model. The research site was chosen because it is a lead user of a highly-customizable software environment, a distributed network of Unix workstations using the X Window System. Data was collected over a period of four months, during which a number of changes occurred, including a reorganization of the staff, a major office change, and a major software upgrade. Both qualitative and quantitative data were collected and I looked for evidence of each of the interactions described in the enhanced version of the Structurational Model, described in Chapter 2. This chapter describes the research site and the events that occurred during the period of the study, the characteristics of the study participants, a description of the data and the procedures for collecting it.

4.2 Research Site: MIT's Project Athena

MIT's Project Athena was created as an "experiment in educational computing" sponsored jointly by Digital Equipment Corporation and IBM. The eight-year, $100 million project has changed how MIT undergraduates interact with computers (Jackson, 1988, Cohen, 1987), resulted in several world-wide software standards (Scheiffler, 1986, Steiner et al., 1988), and influenced the strategic direction of the computer industry (e.g. the creation of the X Consortium (Lampe, 1988) and the Open Software Foundation (Champine, 1987)).

According to the current executive director, (Orcutt, 1990),

> *Project Athena is the world's largest centrally administered distributed computer environment, linking more than 1000 high-performance work stations. Our past accomplishments include the design & development of the "X Window System," a joint project with the X Consortium and Kerberos, an authentication service rapidly becoming an industry standard. Today, we're focused on expanding our delivery of platforms, and on developing the world's most advanced courseware development tools.*

Project Athena was proposed in 1982 and established in 1983 to "improve the quality of education at MIT by providing ubiquitous and high quality computing based on a large network of workstations" (Champine, 1989). Digital and IBM originally contributed $50 million to MIT for a five-year period. Project Athena was granted a three-year extension in 1988 and is currently at the end of the seventh year. The design goals included scalability (up to 10,000 workstations), reliability (24-hours per day), public access (any user can use any workstation to access personal files), secure system services (despite insecure workstations), support for multi-vendor hardware (initially DEC and IBM), software coherence (all applications must run on all workstations), and affordability (with operating costs not to exceed 10% of tuition). In a sense, the efforts of Athena to this point have been designed to build an innovative foundation on which to build innovative education. Project Athena is now a centrally-managed distributed system in which users can access their files from any of the 1000 Athena workstations on the campus.

During the first five years, faculty could apply for grants to support the development of educational software. Over 100 projects were developed for courses. Although no additional support has been given to faculty, many departments have provided their own funds and some faculty have received grants which continue to be used to support course development. At this point, Athena is very much a part of the MIT community. Excluding first semester freshman, over 86% of undergraduates use the system and the percentage is rising. There are over 1100 active user accounts which generate over 4000 logins and about 9000 mail messages a day. Project Athena's management is working with the MIT administration and faculty to evaluate Athena's strengths and weaknesses and propose a strategy for future integration of Athena into MIT's organizational structure.

4.2.1 Organizational structure

Project Athena has undergone several organizational changes over the past seven years. A major organizational change occurred in the middle of the study. The organizational structure shown in Figure 4-1 had been in place since the arrival of the current director in September, 1988 and continued through December, 1989. Two of the top managers left just before the reorganization in January, 1990, shown Figure 4-2.

The new organization has a two-tier management structure with senior and junior managers. Four individual contributors were promoted into management positions and one was promoted from low-level management to senior management. Outsiders were hired for the two most powerful positions: User Services and Systems & Operations. These positions employ the majority of the staff and have the largest influence on the daily operations of Project Athena. Most middle managers were not promoted to senior management positions. The Systems & Operations position combined two previous positions and the total number of senior managers was increased from two

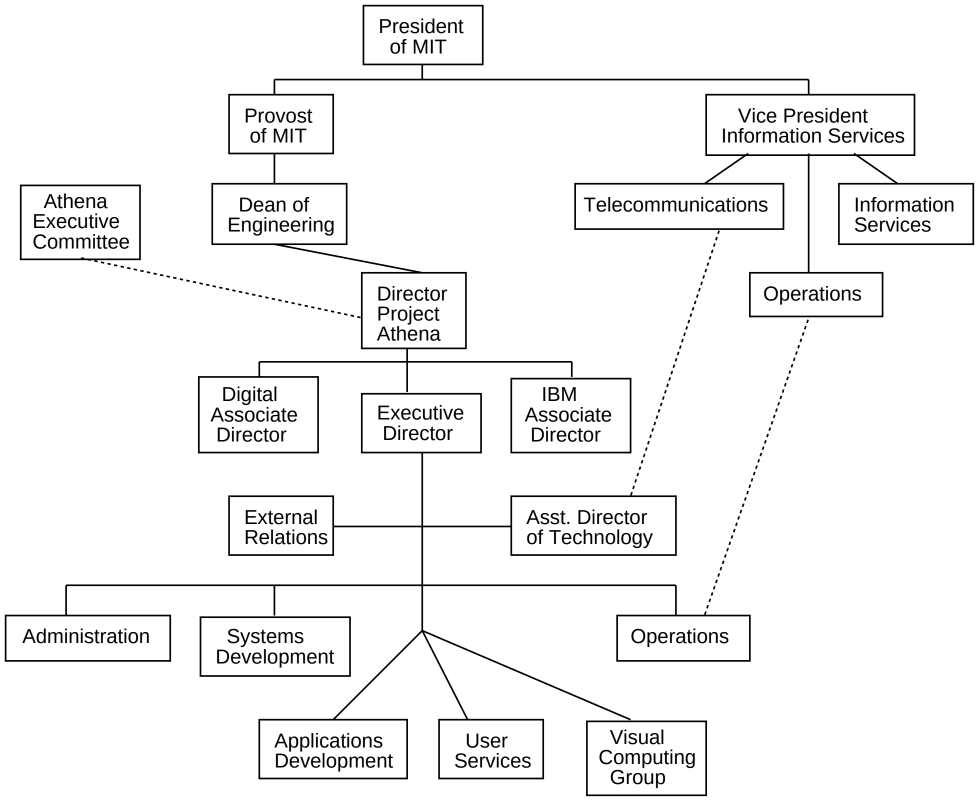


Figure 4-1: Project Athena: Organization through 1989

(both of whom are gone) to seven. Concurrent with the reorganization was a change in offices, which affected approximately half of the staff.

4.2.2 Hardware environment

Project Athena was designed to handle high-performance workstations with a minimum of one megabyte of memory, one million pixels (on the screen), and one million instructions per second (mps). In the early days of the project, these workstations did not exist, so Athena began with a time-sharing model, with attached personal computers. The current system consists of over 1,000 DEC microVAX and IBM RT workstations. Most have 19" black and white monitors, a minimum of 6 megabytes of memory and a 70 megabyte local hard disk, with a mouse and keyboard for input. Additional machines are used for file service and other system management functions. The system is completely distributed; students may log in to Athena workstations located in public "clusters" throughout the university. In addition, some workstations have been placed in dormatories and fraternities and others are located in departmental clusters.

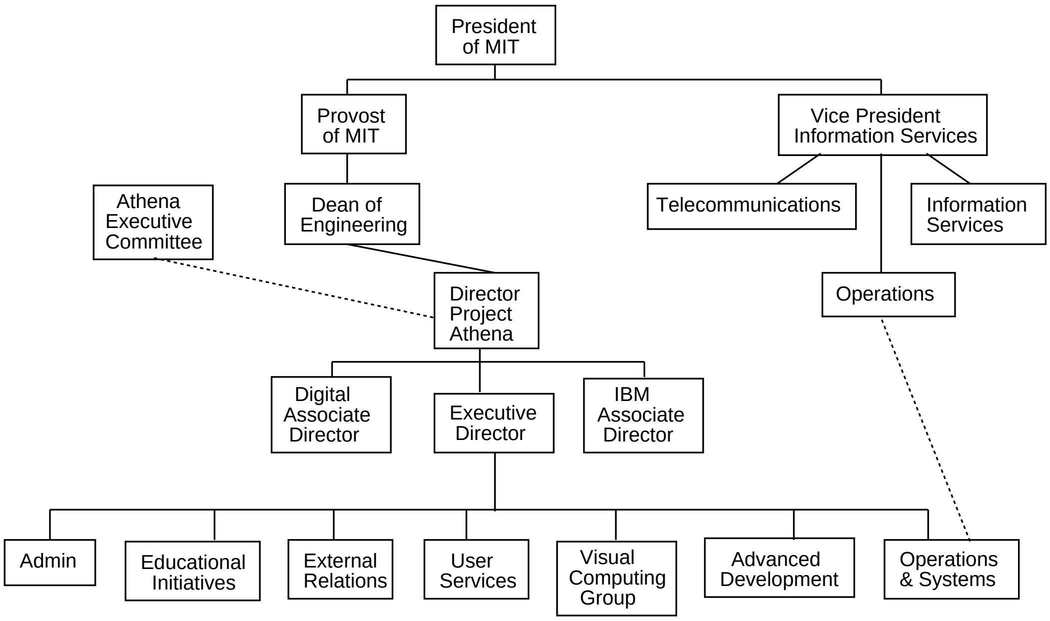


Figure 4-2: Project Athena: Reorganization in January 1990

All members of the Athena staff, from secretaries to managers to programmers have at least one workstation in their offices. (Programmers often have two or three, to enable them to test software on different vendor platforms and increase development efficiency.) These workstations can be made "private" rather than "public". Public workstations are located in public clusters and automatically receive system updates and routine maintenance procedures. Owners of private workstations may choose if or when or to upgrade their software. They may also make modifications to their workstations, from purely aesthetic (such as images for screen backgrounds) to functional (such as enabling the workstation to handle calls from home).

4.2.3 Software environment

Although initially unreliable, the Athena software environment is now quite stable. Major upgrades are tested on the staff first and then introduced at the beginning of each semester and at the beginning of the summer term. Users have individual accounts, can access the internet, and have disk quotas for backed up file storage. Users can easily obtain the password to get local access to any public workstation on campus[7]. However, because of Athena's network security system, users can access only their own files, unless explicitly given permission to access other people's files. In

[7]In Unix parlance, this is referred to as "logging in as root".

addition to their own files, users may "attach" additional file systems and use them as if they were on the local workstation. This provides a very high bandwidth for communication and interaction with others on the system.

Project Athena's software environment is based on a version of the Berkeley Unix operating system. Unix is described by Norman (1986):

> *The underlying philosophy [of Unix] is to provide a number of small, carefully crafted operations that can be combined in a flexible manner under the control of the user to do the task at hand. It is something like a construction set of computational procedures. The mechanisms that make this possible are a consistent data structure and the ability to concatenate programs (via "pipes" and input-output redirection). The interface suffers multiple flaws and is easily made the subject of much ridicule. But the interface has good ideas: aliases, shell scripts, pipes, terminal independence, and an emphasis on shared files and learning by browsing.*

The philosophy of Unix is evident in the software choices made at Project Athena. Users may choose from several text editors (*emacs*, *vi*, *EZ*, *ed*), text formatters (*Scribe*, *LaTeX*, *TeX*, *Nroff*), window managers (*MWM*, *UWM*, *TWM*), mail systems (*mh*, *Xmh*, *rmail*), and *zephyr*, a system for displaying messages on user's screens. Other choices are available as well, but not all are fully supported. Each user also faces choices about how much or little to customize each application and the software environment as a whole. Different applications provide different levels of customization options. For example, the *emacs* editor is the most difficult to learn, but provides the greatest amount of power, flexibility and customization options. Users can rebind function keys, use specialized editing modes, create macros and initialization files, and even write LISP-like programs. *EZ*, on the other hand, is designed to be easy to use, with a WYSIWYG-style[8] interface, but offers limited customization options. A user may try one application for a while and then try another. It should be noted that these applications are used simultaneously which sometimes causes "interaction effects". Also, choice of one application may affect other choices. For example, the *emacs* text editor allows users to use the *rmail* mail system from inside the text editor, rather than using a separate mail system.

Unlike the Apple MacIntosh, which has a unified user interface philosophy that carries over to all (or almost all) of the software on the system, Unix and the X Window System provide users with many

---

[8]WYSIWYG stands for "What you see is what you get".

different forms of interaction with the system. No one person or system has final or complete control over the user interface. Users control some interactions directly, such as with the mouse. Users control other interactions through customization files. Other factors may be controlled by the application designer. For example, an application designer may create a very flexible or a very rigid interface. The designer may also give the user access to control via "X Resources" files or make those decisions arbitrarily. When different applications are executed together, decisions made by those application designers, the window manager and sometimes the system, all interact to affect how the applications work together. System factors may change the user interface as well, or cause "bugs". The window manager sets up default methods of interacting with different applications, which may be overridden by the application, the user, or the system. Figure 4-3 shows the range of factors that influence the look and feel of the user interface for a particular individual at a particular point in time. (Refer to Mackay (1988b) and (Ackerman and Mackay, 1989) for discussions of these competing sources of control.) The combination of these factors makes the system both highly flexible but also sometimes unpredictable. In order to protect themselves, some users seek to minimize confusion by creating standard customizations across applications.

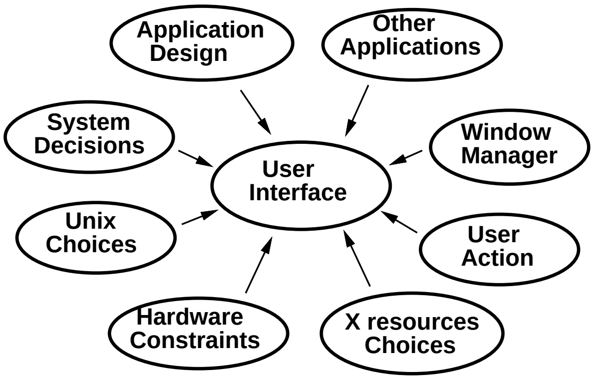


Figure 4-3: Factors that affect the user interface

Applications use the X Window System to display windows on the screen and handle user interaction. Users work in a multi-tasking environment, in which it is possible (and common) to run several independent programs simultaneously. Multi-tasking workstations provide a qualitatively different interaction than single-tasking machines, e.g. personal computers. The difference is similar to that experienced by a typist who moves to a personal computer; not only are the features different (e.g. erasing and moving text), but over time, basic patterns of work change. Similarly, using a multi-tasking workstation, with a large screen and a mouse, often produces a different working style.[9]

Users have a number of different mechanisms for expressing preferences at a number of different levels. The user can specify how an application looks (e.g. font sizes, borders, colors, shapes) and how to interact with it (mouse, key bindings, menus, etc.) Some choices affect all applications, such as the choice of a window manager or the use of Xresources. Others are specific to an application. Customization is generally accomplished by editing a separate file. Because the filenames are usually preceded with a dot, they are referred to as "dot files". Users may be unaware that these files exist, because a normal directory listing does not show them. They can be edited with a text editor and are executed whenever the application is run. Users may exchange parts or all of these files with each other via electronic mail or copying files.

Table 4-1 lists some of the options users have when customizing software in the Athena environment:

Table 4-1: Customization options

"Aliases"	Window layout	Zephyr location
Window size	Window access	Screen layout
Icon behavior	Status indicators	Information access
Key bindings	Screen saver	Screen background
Menus	Fonts	Colors

Some customizations are highly visible and might be noticed by someone walking by, such as an unusual pattern on the background screen. Other items are less noticeable, particularly choices of keys and specifications about process, and will only be noticed if someone is watching the user carefully and notices a difference from the observer’s own use.

Managers are responsible for many of the choices about Athena’s software environment and must operate within a number of technical constraints. They express their decisions by deciding which applications to create, "port" (convert to the Athena environment) and support. They fund the development of particular applications or purchase them from outside vendors, usually in response

[9] If you have not used a multi-tasking workstation, think about working at a desk where it is only possible to work on one thing at a time. In order to work on the next item, you would have to place the first item into a filing cabinet before removing and working on the next item. Now consider working at a desk that allows you to spread out several items at the same time and move smoothly from one to another without changing state. The experience is qualitatively different and the decisions one makes about organizing activities change.

to requests by the faculty. Due to the experimental nature of Project Athena's computing environment, most applications must be customized in order to operate at Athena. (At the very least, they must be changed to accommodate Athena's security system.)

Managers designate software as "supported" or "unsupported". Supported software has training, documentation, consulting and software support. Unsupported software is made accessible by Athena, but support is not guaranteed. Users make an investment when they learn a new software application. One of the factors in their decision is the level of support available and expectations of continuing availability. Managers may change the support status of a software application given pressure from faculty or changes in software licenses.

Management is influenced by the users of the software, mostly the faculty, but students as well. They are also influenced by the technical feasibility of different projects. However, software is often made available on Athena through other sources. Faculty and students in their departments often work on projects for particular courses. Originally, these were funded from a special grant at Athena, but now faculty must obtain their own grants. Another important source of software is SIPB, the Student Information Processing Board. SIPB members are students who like to program and contribute software to the rest of the MIT community. Some of this software is very good; these students know that it will only be used if other people like it and there is tremendous peer pressure to develop innovative programs. These projects often turn into undergraduate theses or research projects. Some of SIPB's "unsupported" software is better supported than the "official" software. The third source of software is the Athena staff. Programmers and other staff members are in a position to see various needs and suggest and develop solutions. Some of these are "underground" efforts, which only become official after the ideas have been tested. (The X Window System and Zephyr are two examples of "underground" systems that became widely used and influential, but only after overcoming internal resistance.)

Project Athena's administrative control over the software environment is intermediate between a centrally-controlled time-sharing environment and a distributed set of personal computers. In the former, an MIS department usually controls access to the hardware and limits the software available to users. In the latter, users may choose from a wide range of commercial software or free software from a local user's group (such as the Boston Computer Society). At Athena, users may install Unix software on their workstations, but with the caveat that many of the advantages of using Athena will not be available (such as system security). Also, Unix does not provide the variety of commercial software available for personal computers.

4.3 Study of customization by Athena staff members

The purpose of the study is to provide an in-depth look at the customization activities of active users of the Athena software environment. Project Athena's staff are interesting to study because they are creating and operating in a work environment that may be typical in 5 to 15 years. The hardware operations staff is tiny: only five people are needed to support over 1000 workstations. Organizations with similar quantities of hardware have three to four times the number of staff. As "lead users" (VonHippel, 1986) of a new model of computing, the members of the staff are first to experience the problems and have the power and resources to make innovative changes. As users of the technology they have created, they are both influenced by it and influence it in turn. They have a vision of the future, but must make it work today.

4.3.1 Participants

The Project Athena staff consists of over 80 people, including managers, secretaries, technical and non-technical staff. The Athena staff provide a variety of services, similar to the MIS department of a large corporation. The director and executive director are responsible for the strategic direction of Project Athena, contact with the MIT administration and faculty, and management of the various activities and services Athena provides. The Operations group is responsible for ordering, deploying and maintaining the hardware maintained, as well as handling emergencies. Systems development is responsible for original code and conversion of third party software to be compatible with the Athena environment. They create a new system release every semester and provide release engineering services. The information officer is responsible for public relations, handling the more than 100 visitors who come to Athena every week. The documentation group writes paper and on-line documents about Athena's supported software. The consulting group provides consulting services to all users of Athena, students, faculty, and staff. The faculty liaison group handles questions from faculty and helps them with course proposals. The visual computing group provides design assistance to faculty interested in multi-media projects and is also responsible for the creation of an authoring language. Administrative staff are responsible for finances, personnel, software licensing contracts, user accounts, non-grant orders, and staff support. In addition, IBM and Digital support a number of full-time staff at Athena and provide an interface back to their respective corporations.

It is important to emphasize that, although they are located at a University, the Athena staff have very real responsibilities and deadlines. A software company may "slip" a hardware or software release date, but MIT never slips the beginning of the semester. An error in the release will affect thousands of people. The MIT community is both forgiving (students graduate, encouraging early mistakes to be forgotten) and critical (members of this community are articulate in criticizing Athena on a number of levels). 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 (Turkle et al., 1989).

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 4-2 lists the numbers of participants from each job category and their technical skill levels.

Table 4-2: Technical background of the study participants

		Technical Level		
Job Category	**N**	**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.

4.3.2 Data

The data consist of open-ended interviews, questionnaires, and automatic records of customization activities. I conducted the interviews over a period of four months. I also held informal discussions about customization with some participants, based on opportunity. 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. The three questionnaires appear in Appendices B, C and D. Questionnaire items include the following:

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

I modeled the latter two questionnaires 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. I also examined the customization files themselves, which often contain a header that identifies the file’s creator and the people who have subsequently modified it. Programmers are more likely than non-technical staff to follow this convention.

I also asked participants to provide the following records:

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 from most recent to least recent.
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).

It is important to note that there are limitations to this data. For example, much of the file sharing that occurs within the organization is informal and does not appear in the on-line record. For example, if one person sees something desirable on another person's screen, she may decide to try a variation of it for a while, and then delete it. This will not show up in the record. Another problem is that people find it difficult to remember the sources of customization files and who the files have been given to, especially if the exchange occurred months or years ago. People tend 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 should underrepresent the level of sharing within the organization.

In addition to these data, I also received information about the organization via the staff mailing list, informal discussions with staff members, a review of the on-line consulting system logs (which include questions about customization), organization charts and the list of office changes.

4.3.3 Interview procedure

The interviews were open-ended, although based on the topics identified in Appendix E. This allowed participants to discuss customization in their own way. I also incorporated new questions, particularly about customization sharing, as they arose and I interviewed several people a second time to incorporate these additional questions. I asked about customization strategies, specific triggers and barriers to customization, and the process by which users make customization decisions.

I set up a one-hour appointment with each participant to discuss the questionnaires and ask open-ended questions about how the participant customizes his or her software environment. 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 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.) Note that many people were unable to remember the names of people they shared customization files with. Thus, these data should thus not be viewed as exhaustive but rather as evidence for the phenomenon of sharing customizations. Additional research is necessary to determine more precisely the actual level of sharing.

Chapter 5

Individual Decisions about Customization

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.[10] While it is theoretically possible for each user to carefully evaluate and optimize each possible customization option, the Information Lens 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. This chapter examines the factors that influence when and how users choose to customize their software and the circumstances that influence their decisions. The effects of these decisions are also examined, both with respect to the user's future behavior and with respect to the organization.

5.1 The problem with customizing: An example

The following anecdote illustrates some of the problems with customizing software. When I write, I often run into an annoying editing error that I could "easily" fix with a customization file. The following are the notes I took on my experience, beginning with a brief description of the software and my use of it.

> *I use emacs, a powerful, customizable text editor with a large number of commands that can be bound to different combinations of keys. Virtually every key combination results in some command, although I use a limited number of them (perhaps 20). As a touch-typist, I prefer emacs to menu-oriented or function-key oriented text editors, which don't let me type as quickly. I have invested quite a bit of time learning it, but now I find it very efficient. I avoid learning new commands, unless I'm performing a new task for which a new command is ideally suited.*

---

[10]Not all end users can make such choices about their software environments. Some must use a single application in a fully-prescribed, standard way. The research described here is relevant to people who have latitude in what, how much and in what way they use software to accomplish their work.

I just ran into a problem that occurs relatively frequently (3 or 4 times per hour, if I'm composing). I hit an unknown combination of keystrokes and find myself in "novice mode". This is always annoying, because it distracts me from what I'm thinking about. Every time it happens, I think, "Why don't I change that key combination so that it simply does nothing, rather than put me in novice mode?" However, the problem is easy to fix (I just type CTRL-G), so I usually just do that and continue, without losing my train of thought.

In thinking about why I don't change the key bindings, I realized that there are several reasons. First, at the particular moment when it happens, I am usually concentrating on something. Although finding myself in novice mode is annoying, because it distracts me, I can easily fix it. My goal is usually to get back to what I was doing as quickly as possible, so that I don't lose my train of thought. So, I never fix it at those times. The rest of the time, I don't remember it. It wouldn't be that hard to fix, but would probably take an hour or two, which I'm never particularly willing to spare.

In order to fix it, I would have to solve two problems. First, I would have to find out which combination of key-strokes causes the problem and its command name (probably something like "novice-mode"). If I can figure out the command, I can figure out the key-strokes from the documentation. Alternatively, I can try to figure out what the combination of keystrokes are (all I know is that it's a mistake made with my left hand and it has a combination of keys) and look up the command in the documentation. I can do this by trial-and-error (I've tried it for a minute or two and given up) or look in the documentation for likely keys.

Second, I would have to figure out the process for changing the key-bindings.[11] *I currently use the "default" set of keybindings, which is located somewhere on the system.*[12] *I would have to figure out how to set up a file with pairs of key-bindings and commands, using the correct syntax, and then get emacs to read it. None of this is hard, and I know where to look in the manual. But it would take me an hour or more. Interestingly, if I do invest the time in setting up the key-bindings file for one command, it*

---

[11] A key-binding refers to the action that is associated with a key or combination of keys on a computer keyboard. For example, in order to move the cursor forward by one word, I might bind a pair of keys, ***CONTROL*** and ***F***, to the function "move-forward-one-word". Alternatively, I might choose to designate a key from the numeric keypad for that function.

[12] This is a distributed Unix system, in which there are invisible default settings which can be overridden with a user customization file.

should be easier to change other key combinations. If I do this, perhaps I'll also fix the other problems, which occur less often but would be relatively easier to fix.

At this point, I decided to try to fix it, to see if my intuitions about the difficulty of customizing were correct.

I just tried each of the strategies I mentioned earlier. I haven't been able to find "novice" in the documentation under any of the categories: key index, command index, variable index, concept index. Nor have I been able to replicate my set of keystrokes. I tried doing it systematically and ended up in several strange modes and then exited unintentionally. I have also been unable to find a likely candidate by scanning the list of keys.

I've looked in the table of contents and found "Customization: Customizing Key Bindings". It says I should be able to change "key maps" in my .emacs file or with a command. It gives me the syntax as: (char . binding), but no examples. I've tried several of the commands listed (ctl-x-map), but none seem to work. I give up and exit.

I just ran emacs again and it's in a very strange state. Only half of the window works. I get out again and back in again. Same problem. None of the window control commands I know have any useful effect. This is the other reason I don't customize as much as I'd like: It's too dangerous!

A friend stops by to chat. I ask her if she's ever seen anything like this before. She says no, but she'll ask a friend of hers. The friend isn't there, but another person tells me to try CTRL-X O. I'm not sure why this would be helpful, but try it anyway. Nothing happens. In disgust, I exit. I immediately run emacs again and now everything is normal again. Did that fix it? Who knows.

The story illustrates a number of the issues users face when deciding whether or not to customize their software. First, the cost of making the customization change may be too high at the point when it is most relevant. As predicted by Repair Theory (Brown and Van Lehn, 1980, Burton and Brown, 1979), since the problem could be easily repaired, it did not represent a significant breakdown for me. I did not deem it worth fixing, despite the annoyance factor. Second, customizing is not always a benign activity: users learn that customizing may cause things to break. Knowing this, many users avoid making changes, preferring to do without or looking for customizations that have already been tested by their friends. Also, one's estimates about what is involved in fixing the problem may be quite different from the actual amount of time required, yet it is the perception of the cost that affects the user's choice. In this example, my estimates were quite far off, both in the identification of the command and the difficulty of finding a solution. I have now increased my estimates of how long such customizations should take, which will influence my future decisions about customizing. Finally, the example shows that certain kinds of external events can encourage the user to take the extra time to customize. I only took the time to customize when an external event, writing a chapter about the decision process of customizing, caused me to think about the issue. I was reminded of the problem as I was writing, but only devoted the time to customizing because I was reflecting on my own use of the software, could afford the distraction and was curious about whether or not my estimates about fixing it were correct. This implies that one way to encourage people to customize is to set up circumstances in which they both can afford the time and are paying attention to the problem.

5.2 Case studies of customization

The members of Project Athena's staff customize their software for a variety of reasons, under a variety of circumstances. The purpose of this section is to illustrate both the diversity of their approaches to customization and to identify some of the common patterns that they share. I have chosen five individuals, one from each of the major job categories: manager, secretary, systems programmer, applications programmer and other staff. After describing the characteristics of the people in that job category, I present the specific customization (or lack of customization) activities of an individual from that job category, as well as the individual's technical skills and role in the group. Because customization decisions are based on *perceived* rather than *actual* costs and benefits, users' perceptions of their own behavior are as important as their actual behavior. Where possible, I compare the individual's perceptions about their customization activities with their actual behavior.

5.2.1 A non-technical manager

The ten managers in this study represent a wide range of backgrounds, from an extremely technical programmer who made significant contributions to the underlying system architecture (A1) to a completely non-technical former artist who is completely unfamiliar with the technology (A8). They are responsible for very different functions in the organization, but share a high "people" component in their work, and use the technology to interact frequently and effectively with their staffs and their peers. Managers A1, A5 and A6 have programming backgrounds, while the rest do not. Most of the managers avoid customizing, except for A1, who was recently promoted to management, and A3 and A6, who do it when they are bored or want to explore the system. They cite a variety of reasons but the most common are lack of time and a desire to use the system that the majority of the users use. The most common kind of customization that they *do* perform is to encode a commonly repeated pattern of behavior, which is valued because it increases productivity but doesn't create a fundamentally different experience of use.

A2 has the distinction of having been at Project Athena for the longest amount of time, over six years. She has two supervisors reporting to her, A3 and A4, and is responsible for consulting at Project Athena. Her group is part of the User Services organization, whose members are responsible for answering questions from Athena users, either in person in the workstation clusters or through the on-line consulting system (OLC).

A2 rarely changes her customization files and some date back to 1984. "I let the clemantis vine grow forever!" The last major set of changes she made were the result of the Fall software release, which introduced a new method of organizing customization files. In the intervening nine months, the only subsequent changes she has made are to update her aliases[13] file and her Zephyr subscriptions[14] files. She uses about 80% of the aliases in her aliases file, and adjusts her zephyr subscriptions regularly. Although she knows it's possible, she hasn't made any other kinds of changes. "I haven't bothered to do anything with the format."

She does not use a standard screen set up when she logs in. She has a procedure that she goes through whenever she logs in, in which she sets up her aliases, attaches the relevant directories and sets up her windows. (The first two steps are performed by running a file. The last step involves individually setting up windows on the screen.) Since she logs in once a day, this is somewhat cumbersome. However, she does not view it as a problem, except for her observation that she has to devote mental energy to doing it when she would rather be thinking about something else. "I'm quick and it takes less than a minute to log in. But more important than the time is that it wastes mental attention."

"If I need to be more efficient, I do individual attaches."[15] She says she doesn't do it automatically

---

[13]Aliases allow users to specify a shorthand for a more complex term, command or set of commands.

[14]Zephyr is a communication system that allows people to pop up messages on each other's screens. The subscriptions file is similar to a distribution list in electronic mail.

[15]Users can "attach" other directories to their current directory tree, which allows them to treat files from other parts of the network as if they were local. With appropriate permissions, users may attach other people's home directories or file systems, such as the Student Information Processing Board's set of software.

because "I never figured out how." Part of her problem is that the procedures for doing this have evolved and she has not kept up with the changes. She has also developed a set of habits that were appropriate when the Athena software was less stable but are inefficient now. For example, she says, "I try to logout every day and reboot every four days." In contrast, many other staff members stay logged in for weeks and sometimes months at a time. E15 says, "I never log out -- I stay logged in for two to four weeks." C9 claims he hasn't logged out since the last system upgrade. Logging in daily is especially inefficient for A2, because she has not automated the process and must do it manually every time. However, because she has no evidence that the underlying problems have gone away, she has no incentive to change what is now essentially superstitious behavior.

She would like to customize more, but feels that her other job responsibilities occupy too much of her time and energy. She says, "The biggest barrier to customizing is finding the time to do it. Life at Athena is like page thrashing. I handle personnel first, otherwise I die, then administrative things. I have no time to read. It's amazing what I don't have time to do." She is quite aware of what the system can do, since she is familiar both with her staff's work environments and the kinds of questions that users ask. She gave several examples of customizations she'd like, such as setting up Zephyr to give her automatic reminders about meetings and other events.

A2 does not have a standard working environment, since it has evolved as the system has evolved. She has never taken the time to start over (which would be easy for her since she has very few active customizations) and use the current standard. Over the years, she has lost many of her original customizations. She gave an example of a customization that she liked which allowed her to keep drafts of mail messages. "I liked drafts. [During one upgrade], it broke and I set it aside. Maybe it works again -- but it's a relic." (E16 has a working version of this, but she is not aware that it is available.)

A2 suffers from having been in the environment for a long time, during a period of extreme change. Because she is so busy, she has never taken the time to reestablish a useful set of customizations. She tries to buffer herself from change by performing many actions by hand. When asked what would help her, she said: "It depends on time efficiency. If one hour could set me up, I'd set up a time in the next three weeks. I'd like a customization service. I'd order it and see if I could maintain it." However, she is very nervous about getting things that she won't remember how to maintain. She receives no feedback about the inefficiency of her use of the system or how to improve it. Nor does she have any feedback that tells her that certain time-consuming actions, such as logging in each day, are no longer necessary. Her approach to customization is extremely conservative and her previous history continues to affect her day-to-day use, often in non-productive ways.

5.2.2 A non-technical secretary

The five secretaries are more similar than the managers in their use of the technology. With the exception of B1, all are relatively new to Athena and none have programming or strong technical backgrounds. None of them have customized their software much, and when they have, it has been with the help of others, primarily E2, A3 and A10, their manager.

B1 is fairly typical of the secretaries. She has been a secretary at Project Athena for almost three years and is responsible for providing secretarial services to the general staff. (B3 and B5 are also "general" secretaries, while B2 and B4 are secretaries for the director and executive director of Athena.) B1 uses the software in a very constrained way and is largely unaware of the features of the technology. She does not know what a window manager is and was not aware that she had any choice about it when the window manager changed in the system upgrade. She never changes any of her own files, but people sometimes help her and set things up for her. She uses the screen setup that A10 gave her, which comes up automatically when she logs in. When she wanted to change the arrangement of windows on her screen, she got help from A3 and E16 (about six weeks ago). She is not aware that she doesn’t have to log out everyday, nor is she aware that she can set up an aliases file for things she types frequently.

The primary customization trigger for her is when something breaks or changes unexpectedly. "If there’s a problem, I ask someone." She gave a recent example in which her editor was suddenly filled with odd characters. She asked A3, who was able to fix it for her. When asked about how well she likes her current set of customizations she said, "It’s all right. Especially now that the clock and calculator are in the right places. However, the windows are black and I’d prefer white. I wouldn’t move things or put up anything else." She is basically unaware of what is possible, and although she asks for help periodically, she doesn’t get everything she wants. B1 is completely at the mercy of the system and knows it. She is unaware of the possibilities and cannot fix things if someone tries to help her and ends up breaking something.

5.2.3 A very technical systems programmer

The twelve systems programmers can be divided into roughly two groups. The "young" group are former MIT undergraduates in computer science who learned about Athena and contributed to its development over the past few years (C1 - C7). Although in their early to mid twenties, they generally cite 8-10 years of computer experience each, indicating that they have been fascinated with and working with computers throughout their adolescence and early adulthood. All of these people maintain extensively customized software environments and make their customizations available to others in the organization, primarily by giving other people open access to their files. They pride themselves on their technical prowess and their customizations are almost a form of competition, to see who is the cleverest. Reputations are very important to the members of this group and their impressions of themselves are very much affected by what their peers in this group think. They also feel that they are at the top of the hierarchy at Athena, being the most technically talented. Interestingly, this is a hold-over from the early days of Athena, in which technical skills were the most highly prized. The new management is much more concerned with making Athena into an effective and efficient service and the new people who have been hired reflect that objective. Many people have left the organization and some of the remaining systems programmers are a bit bewildered by their sudden loss of status. They have been split into different groups and have been discouraged from making any major systems innovations. Their role is now one of support for an on-going service operation rather than creating a new kind of computing environment.

The second group of systems programmers are significantly older (30-45) and have industry experience outside of Athena (C8 - C12). Although technically as skilled as the first group, they no longer find the system fascinating as an object in itself. All pride themselves on their professionalism and they feel that they are being paid to do a job, not explore the system. Many of them refer to the first group as "kids" and some admit that they were once like them. This group has all of the necessary technical skills to significantly customize their work environments. What is interesting is that they, for the most part, choose not to customize. Section 6.3.4, in Chapter 6 discusses this in greater detail.

C5 is a member of the first group of systems programmers and customizes his work environment extensively. A former MIT computer science graduate, he has been working with Athena since 1986. He has three "privatized" workstations in his office with three different hardware architectures (an IBM reduced-instruction set (RISC) workstation, A DEC RISC workstation and a DEC VAX). He uses a single monitor to display windows from all three workstations at the same time. He has avoided the general system upgrades and adds new components of the system software incrementally, when he finds the time to do it.

C5 started with XWM[16] when he first arrived and then converted to UWM, which he customized extensively. He also experimented with a few other window managers. He said he avoided one of

[16]Athena has provided a number of different window managers over the years. XWM (The X Window System Window Manager) was the first default window manager at Athena, but was later replaced by one called UWM. MWM, the Motif Window Manager released by the Open Software Foundation, is the most recent.

them, GWM, because "It was too big to put all my customizations in." He converted from UWM to MWM, but it took over a week. "I started with the standard default for MWM and modified it over time. I deleted the non-working stuff, but kept most of the changes." He described his overall strategy when moving from one window manager to another: "I keep the menus the same when there's a cross-over. When it's stable, I do further customizations." He uses menus as his primary mechanism for getting to functions. "I don't use the keys for much of anything, except to get to the menus." He plans to use TWM at some point but has heard about some bugs, so he's avoiding it until they are fixed.

C5 has the technical skills to avoid using the customization mechanisms provided in the code. For example, instead of using Zephyr's customization files, he "replaced the initialization sequence with my own shell script." He also customized the early version of the window gram[17], and then redid it when the new version came out. He said this was easy because "it was conceptually similar." He did not ask anyone about how to do any of this. Instead, he read other people's files and the source code. "I play with it until it's close -- then I keep going until it's right."

C5 says he customizes "about once or twice a month", which he feels is a relatively low level. In fact, this is an average level compared to the other young systems programmers and a very high level with respect to everyone else. "I have a lot of customizations, but I don't change them very often. I usually only do it in response to some problem. For example, I changed my .Xresources file to get the backgrounds 'just right'. It changed because the [DEC RISC workstation] runs slightly different software." He then proceeded to explain the details of why different applications "didn't understand" the .Xresources file.

C5 has a set of aliases but doesn't use most of them. "Of 30, I only use 6." To handle the functions that most people use aliases for, he has an interesting innovation. He saves the last 1000 commands in his history file[18] and uses the up arrow key to scroll through them to find certain commands. "The advantage of having a saved history of 1000 commands is that I often [need to] type the ones hidden in there." He says that he'll remember that he typed a command that was similar to one he needs now. "I may have typed it 200 commands ago, but I know it's still there. I don't need it for long, so I don't make an alias. I just keep it around for now." Because the history

---

[17]A windowgram is the name of the message that Zephyr pops up on the screen.

[18]The history file saves a record of previously typed Unix commands. The user can specify how many to save, and can set a variable that permits scrolling through and editing these files with the arrow keys on the keyboard.

file is occasionally deleted, he has a routine in his .logout file to check if the history file is empty. If it is, he replaces it with an old history file, containing 1000 entries. He gave an example of how he used it recently: "I attached a file system [with a particular command] about 200 commands ago. I know [the command] is still there. I don't need it for long, so I don't make an alias. But this allows me to keep it around for now."[19]

When he decides to customize, C5 says, "I usually try to figure out how to do some specific thing." He also customizes in the course of doing something else. "Occasionally, when I'm looking for something else, I may find something. For example, I tried to figure out the window gram. I was looking at something else, saw C1's and wanted a similar effect. I spent several hours working on a more general implementation, but it's only half done...I was curious, it looked interesting, and it was interesting to do. I may get back to it, but probably not. I haven't found a need for it."

For him, the biggest barrier to customizing is "mostly lack of interest. I only do it when I feel a need for it. So far, I haven't felt a need to bypass the existing set up." He explained how he manages his growing list of customization files. "I try to keep my home directory from filling up with dot files. I set up subdirectories, but that takes an effort to make the program look elsewhere [and find the customization files]." He pointed out that the office change had no effect on any of his customizations, because his working environment didn't change. "I only had to change my office registration."

As a systems programmer, he is technically capable of making any customizations he likes. Although he has customized his work environment extensively, he does not feel that he has enough time to do everything he'd like. He views customization as one of the ways in which he learns about the system. He often enjoys the process of customizing, although he'll only spend the time to finish something if he feels that he really needs it.

5.2.4 A novice applications programmer

Four of the eight applications programmers in this study are members of the video group, which has a number of special characteristics, described further in Section 6.2.1 in Chapter 6. All of the remaining applications programmers were formerly MIT students although only one completed a degree in computer science. All of the applications programmers are interested in creating useful

---

[19] A1 independently invented a similar use of history files, which he uses "to replace alias". He also sets his history file to 1000. "Then I can use it as a model for new commands by using the lineeditor. It's a good way to parameterize complex commands."

applications for students and other non-technical users. They feel that the systems programmers are much too interested in the system itself. (Only D5 and D7 are technically skilled enough to be systems programmers.) Group members vary in their attitudes toward customization, ranging from little interest (D1) to very extensive customizations (D7).

D1 was formerly a member of the operations group where he was responsible for setting up and fixing broken hardware. He showed a talent for using the system and helped the other members of the operations group get set up. For example, he created an initial login sequence for each of them, which brings up all of the applications they use, particularly the program that tracks incoming hotline calls. The former manager of systems development encouraged D1 to learn to be a programmer. In the reorganization, he was moved into the Educational Initiatives group and is working on providing a common mechanism for Athena users to customize their workstations. D1 enjoys working with others, and still helps some of the members of his old group. However, as the least technical applications programmer, he does not help his new group much.

D1 uses a standard screen set up which "evolved over time." He says a large portion of his work is using Unix shell commands and electronic mail. "This setup will change when I'm looking at more source code." He uses Zephyr a great deal and says "I customize it a lot." He's made it so that personal messages show up on a different corner of the screen: "It lets me differentiate without having to read the message." He got the actual software from another staff member, but says: "the idea was mine."

D1 says "I intentionally didn't customize much -- it used to be my job to know how users were losing. So I didn't want to be different. I use the 'common [Athena] thing' -- it works for me." When he does customize, it's usually because he recognizes a repeated pattern: "I perform an action so often that I set up a default or a one-word command." He has 18 aliases and uses them all. "When I type something a lot, I create an alias. I also use aliases for common typos." (Note that a number of others, including A3, A4, C2 and E13 also use aliases in this way.] He cleans up his files on a regular basis: "I delete aliases about twice a year." He also says that customization is not something he does when he's bored: "I only use it for productivity. I don't play computer games either."

D1 does not feel that there are any particular barriers to customizing: "I customize as much as I want. I just don't think it's necessary to do more. If I think that changing something would help me work, then I make the change. If I don't know how, I can easily ask someone." When in doubt, "I look at other people's [customization files]. That makes it harder to mess up a file."

D1’s favorite innovation was his background screen, a video image of his girlfriend. Whenin the operations group, he used to repair the multi-media workstations. One day, he was checking to make sure that the workstation could take a video input signal from a video camera and display the image on the screen. He realized that the video image had been digitized as part of this process and he discovered he could capture a single frame of video. He investigated how to save a black and white version of this image into a file. Once he figured this out, he went through several more steps and was able to create the image in a form that could be used as a full-screen background image. He was very pleased with this innovation, both because he earned the respect of the systems programmers and it was the first thing that convinced him that he could actually do interesting things with the software himself. When he had done this, he set up a sort of ’video studio’ with the camera and video workstation and invited people to sit in front of the camera and take their own snapshots. The success of this venture, which did not involve any programming but took advantage of a number of different system features and customization techniques, convinced him that he would like to become a programmer. Unfortunately, by the second interview, he had removed the video image and had returned to using a plain gray background. It turned out that he had recently broken up with his girlfriend and hadn’t come up with a new image. As he pointed out: "You don’t make just anyone your root window."

5.2.5 A non-technical staff member

The sixteen "other staff" members make up the most diverse group in the study. They perform jobs that do not require programming, including: running the visitors center (E12) and working with faculty (E1), training (E2), release engineering (E3), documentation (E4, E11), network maintenance (E14), consulting (E16), operations (E6, E9, E10, E13, E14), consulting with faculty on the design of multi-media applications (E5, E7), and maintaining user accounts (E8). Most do not have technical backgrounds. Only E14 was a computer science major with a strong background in computing. Some are former English majors (e.g. documentation writers), others have artistic training (e.g. the visual computing group), and still others have not completed or gone to college at all (e.g. the operations group).

E10 is a new member of the hardware operations staff. He is responsible for responding to requests for things like making tapes, changing privileges, etc. He is not very experienced with the system, but has used an IBM pc before and is eager to learn. "I use the computer mostly as a communication tool. I keep memos, discuss meetings, tape requests, etc." He feels that "the communication aspects of Athena are really nice."

When he first arrived, E13 helped him get set up by giving E10 his files. "E13 spent several hours with me -- he was eager to give me his files...On the first day, E13 just said 'here, take it.' Then I just edited it." He found E13's files somewhat overwhelming, especially his Zephyr subscriptions. "E13 subscribes to everyone -- he gets 60-70 messages on his screen. So I had to pare it down."

He has made some customizations but is resisting doing more until he learns more about the system: "I never use any aliases, I'm trying to force myself to learn the standard commands. I don't want to start to customize until I learn the commands...Now that I know things, I've learned the keyboard commands -- I'm happier." When he does customize, he tries to avoid major changes. "Usually the screen things are cosmetic."

He is curious about the system. Because of the design of the system, he can "attach" other people's home directories, if they give the appropriate permission, and have access to the files as if they were in his local directory tree. "I wander around to find something. I go junk hunting. I don't do it a lot, because I get lost. [If I find something, I usually don't copy it], unless it's really neat. I just discover dumb things." He describes what he does when he discovers something interesting: "When I see something neat, I see the path to it, attach it, and try it. If it's good enough, I keep it." He has plans for learning more about the system when he has time. "If it have to do something many times, I'll make a cshell for it. I will learn awk[20] someday." He also plans to learn the programming language C.

He says he customizes to make things neat. "I hacked the login screen so that it doesn't say 'Welcome to Athena'. I like things out of the way. I don't like clutter on the screen. If there are too many windows, I get lost." He cites lack of knowledge as the biggest barrier to customizing. He tries to get around this by talking to people and trying out their ideas.

As a newcomer to Athena, he uses customization as a way to discover more about the system. Unlike most of the members of this group, he enjoys customizing. He has chosen to avoid encoding common patterns of behavior, because he wants to make sure he knows and understands the "standard system". However, the other forms of customization provide him with a socially-acceptable way to explore the capabilities of the system and to learn how to learn about it.

[20]"awk" is a Unix sublanguage designed for scanning and processing patterns in text.

5.3 The process of deciding to customize

What factors influence a user's decision whether or not to customize? The co-adaptive model suggests that a variety of factors, including organizational and group norms, external events, the user's perception of the technology and the user's past customization choices may all have an effect. At any point in time, users have a choice: should I customize the software or customize myself? According to A2, based on observation of students by consultants in the workstation clusters, some people prefer the latter. "They prepare personal cheat sheets, thus effectively customizing themselves rather than the software for the uses of the software that they typically make." While this may be cumbersome, it has the advantage of posing few risks. Even though if everything goes right, the actual time spent customizing will be quite small, many users refuse to do things that may cause problems and end up taking a long period of time. A5 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 spend an unknown time making the change. (The only thing that's known is that the time will be longer than predicted.) Users risk causing other things to break and also risk spending an hour or more doing something that will eventually not work. Previous successes and failures in customizing affect the decisions here. Note that seeking help from others helps to reduce this risk. If someone else has already invested the time to make something work, then the chances that it will break are much smaller.

Figure 5-1 represents the decision process users follow when learning to use a new software application and deciding whether or not to customize it. The first decision is whether or not to even try it. If the application promises a significant improvement over an existing application, but requires a significant amount of 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. Other factors will also affect the user's future customization decisions, including social pressure and other influences from the organization, external events, internal factors and the characteristics of the technology itself.

Users initiate changes, innovate, or reorganize their work environments for a variety of reasons. Some are reactive, based on changes in the external environment. Examples include changes in the software environment itself, such as an upgrade to existing software or the addition of a new

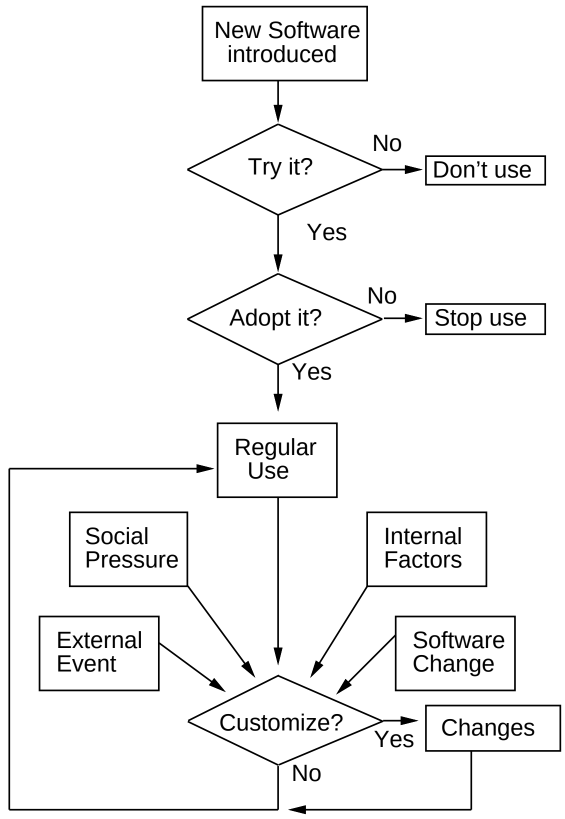


Figure 5-1: Decision process for customizing software

program, and changes in the work environment, such as new job tasks or going on a trip and changes in the physical environment, such as a new office or a natural disaster (e.g. an earthquake). Other changes are proactive. Users may decide to organize or customize their software or learn a new program because of boredom, spare time (e.g. a three-day weekend or an evening in a hotel room while on a trip), a new idea (e.g. seeing something someone else did and wanting to try it, or coming up with a new way of doing something), or through frustration about something.

5.3.1 The influence of external events

Software upgrades are sometimes benign, and do not require the user to modify anything. This is most true for people who have made very few customizations. However, the people who do make customizations in this environment must often make adjustments when there is a major system upgrade. Figure 5-2 shows a representation of typical strategies for customizing over time, in the face of changes in the system software. Note that the patterns of changes are very different according to the kind of customization. Customizations that involve encoding common patterns of behavior tend to increase over time and are not particularly correlated with changes in the system (except to the effect that changes the likelihood that the user will repeat a particular pattern of behavior). These kinds of customizations are least likely when the user is learning the system, because he or she is not yet aware of which patterns are most likely to be repeated. In contrast, system upgrades that affect low-level learned skills, just as the use of keys or mouse buttons for particular functions, are very likely to be affected by system changes. In this environment, everyone who had customized their window manager retrofitted the window manager in some way when the system was upgraded (or avoided changing window managers). The most common form of customization here was to change key bindings so that they continued to act ’normally’ for that user. Customizations that are made as part of the user’s initial explorations of the system are most likely to be dropped with system upgrades that affect them. Even though they are considered "neat", they are usually not essential, and if the system change causes them to break, most users will delete them rather than try to retrofit them so that they continue to work.

These data indicate that people react to forced changes in customizable software in one of three ways: they avoid customization and accept the new changes (sometimes willingly, sometimes not), they customize extensively and retrofit successive generations of the software to be like earlier versions, or they reflect on their previous use of the technology and innovate. Innovation and stagnation appear to be two different reactions to change. Some people respond by trying something new, but most respond by trying to change the environment to be more familiar, as it was in the past. Additional research is necessary to determine the kinds of conditions under which each reaction may occur. The results presented here suggest that people who innovate: 1) have some familiarity with the technology but are not long-standing users, 2) are not overwhelmed by the change, and 3) have opportunities to observe and interact with others who use the technology in slightly different ways.

The members of the Athena staff experienced a major reorganization in early part of this study. Some people changed job responsibilities, either by being promoted (A1 and A7) or by moving into a new group (most of the rest of the staff). A1 denied that his job change affected his work. (He moved from being a senior technical staff member to a manager with two people reporting to him, so the personnel time requirements did not increase very much.) However, A6, who also moved from a programmer position to a management position (although before this reorganization), says: "My current job requires no programming. I would configure it differently if I did. Now I mostly just read mail and respond to it."

D1 is just learning to program for his new job. "A large portion of my work is still shell commands

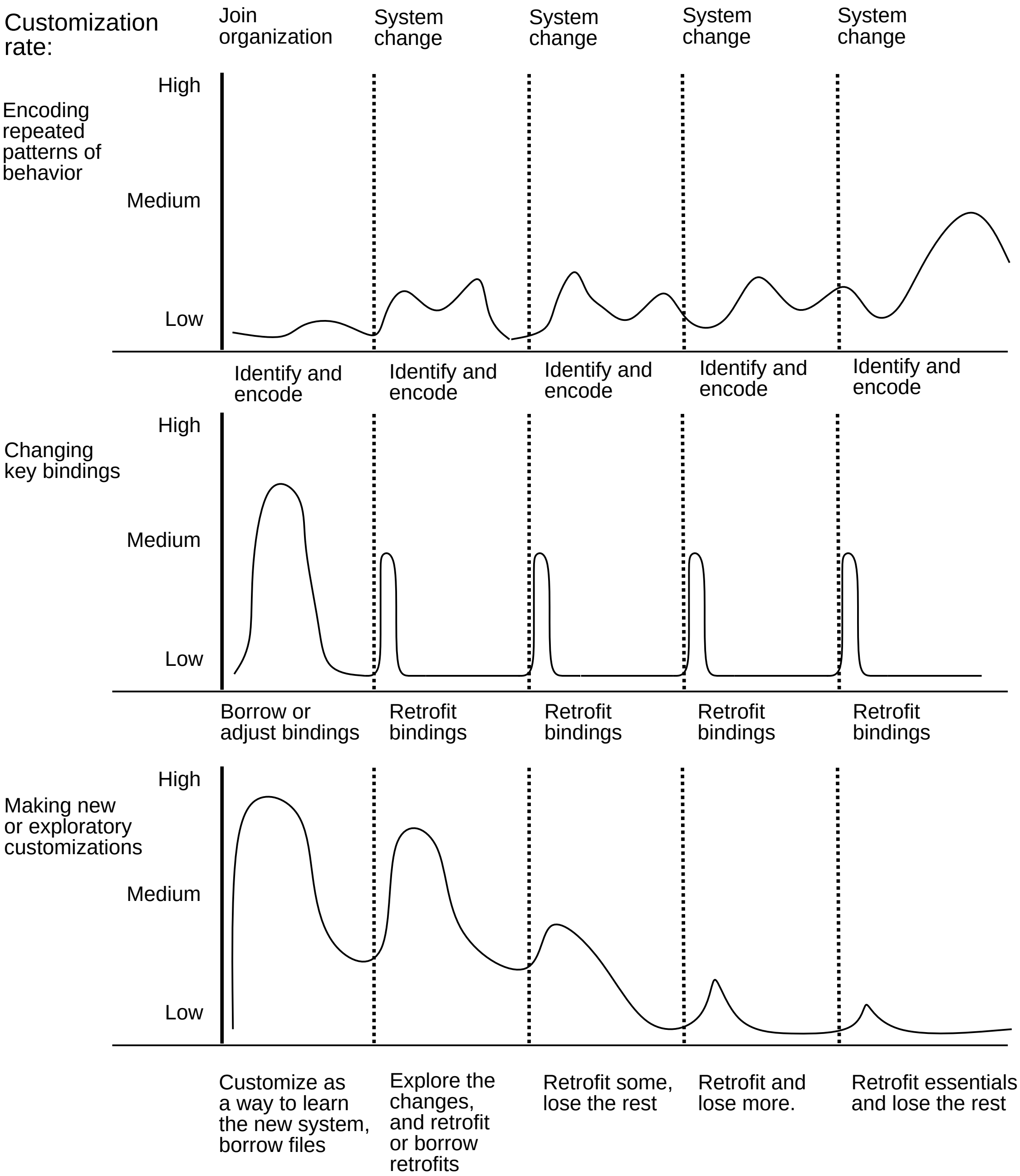


Figure 5-2: Effects of system changes on different customizations

and email. This setup will change when I'm looking at more source code." He uses D8's system for creating two screen setups to handle his former work style and his new work style. C8's job responsibilities now require him to spend more time writing code. "In my debugging environment, I need lots of concurrent windows." He describes his work at another site: "I just use a terminal."

The members of the staff also experienced a major office move, near the end of the study. People who share offices often, but not always, share customization files. Figure 6-5 in Chapter 6 shows which staff members shared customization files both before and after the office reorganization. There is no evidence that the actual move from one office to another increased customization activity. However, in a few cases, it changed the person mostly likely to be asked for customization help and therefore indirectly affected user’s customizations.

5.3.2 Triggers and barriers to customization

Participants were asked an open-ended question about which factors trigger their customization activity. Table 5-1 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 arrows in the structurational model that influence a user’s decisions: the technology itself (arrow B), the properties of the institution (arrow C), external events (arrow F), and individual factors (arrow G). The factors are listed from most to least common in each category, preceded by the percentage of study participants who cited the factor. Note that these data were compiled from an 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 the ones that were relevant.

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 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: 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 the repetition.) 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).

Barriers: By far the biggest barrier is lack of time, cited by almost two thirds of the participants

Table 5-1: Factors cited as triggers and barriers to customization

Percent of Users	Triggers of Customization	Percent of Users	Barriers to Customization
Arrow B: 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
Arrow C: Organization/Others 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		
Arrow F: 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		
Arrow G: Individual factors influence user			
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
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

(63%). 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.

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 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 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' 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 are retrofitted 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.

5.4 Who customizes?

Participants in this study exhibited a wide range of technical skills, from non-existent to highly skilled. It should not be surprising that more technical people were better equipped to customize than less technical people. However, it is not the case that technical people customize and non-technical people do not. The non-technical staff exhibit a range of technical ability, from the secretaries who do not customize at all to E13, who customizes constantly. Similarly, the technical staff also exhibit a range of customization activity. The young systems programmers customize extensively, while the older, most senior programmers customize little or not at all. (This was also true of the Lens study, in which the most complex rules were written by programmers and non-technical secretaries.) Some users find the system overwhelming. E14 says "It boggles my mind". Others spend inordinate amounts of time customizing it. C7 says: "I have spend more hours on my own productivity tools than on the product." He adds: "I have seldom analyzed if it is worth it."

5.5 Common patterns of customization

The following sections identify some of the common patterns of customization that recur among members of the Athena staff.

5.5.1 Exploring the system

Many users customize as a way to explore and learn about the system. Users are most likely to explore when the system is new, either because the user is new to the organization or because the software has recently been introduced. Most users settle upon a pattern of use, influenced by the customizations they've already made, and stop further exploration through customization. A few individuals continue to explore the system, even after they have been using it for a long period of time. These people talk about periodically getting bored with their current set of preferences or using customization as an interesting way to occupy time while waiting for something else. In the Athena study, a number of people expressed this on-going desire to experiment, including managers, the younger group of system programmers, two applications programmers, and a few of the other staff members. A few individuals, notably D, F, and especially G also continued to explore the use of the system over a period of a year to a year and a half.

When a new employee joins the organization, he or she usually borrows a set of files from a co-worker, usually a member of the same group, and trys to make them work. If the person has no technical skills, very little further exploration occurs. (The exceptions are to make 'cosmetic' changes, such as modifying the background pattern or changing colors.) All of the secretaries except B5, managers A7 and A8, and staff members E5, E6, E7 and E12 fit this pattern.

Many people with more computer experience (although not necessarily programming backgrounds) start out in the same way, but then spend time modifying the files they have received. When they run into problems or see things they are interested in learning, they ask other staff members questions (and quickly learn who has the most useful advice). This strategy allows them to learn about the system, both its capabilities and how it can be changed, and lets them set up an environment that they can become comfortable with. After an initial period of exploration, these people tend to settle down with a fairly stable environment and only customize further to a) encode commonly-repeated patterns of behavior or b) maintain the environment when external forces cause it to change. When it becomes necessary to learn a new software application, they engage in a similar procedure, borrowing files and asking questions until they understand. The people who fit this category include managers A2, A4, A9, and A10, one secretary, B5, most of the applications programmers, D1, D2, D3, D4, D5, and D6, and most of the remaining "other staff", E1, E3, E4, E9, E11, and E14.

Some people continue to actively explore the system even after they have set up a comfortable working environment. While these people are more likely to have a programming background, not all of them do. They talk about getting bored with the current environment or enjoying the process of customizing. They like to keep learning about the system although only a few of the younger systems programmers feel they can afford to spend much time doing this. This group includes: three managers (A1, A3 and A6), the younger group of system programmers (C1, C2, C3, C4, C5, C6, C7), two applications programmers (D7 and D8) and a few of the other staff members (E2, E8, E10, E13, E15, E16).

A few people say they never customize as a way to explore, even when they are just learning a new system. These people are all experienced former or current programmers and learn only what they need to get their jobs done. This group includes A5 and the older systems programmers, C8, C9, C10, C11, and C12.

5.5.2 Expressing individual preferences

One of the intuitive reasons for customizing is to create a unique environment that is particularly suited to an individual. In this study, expressing individual preferences and creativity was a common, but not universal customization goal. The modes of personal expression were almost exclusively related to the visual appearance of the screen. Some items involve trade-offs. For example, dark windows with light text reduce flicker but lower contrast, whereas light windows with dark text increase flicker but also increase contrast. Different individuals are more sensitive to one or the other and make the trade-offs according. D2 says, "My policy is to eliminate any pixels I can, because I'm sensitive to flicker." People with color screens can make much more subtle adjustments and are more likely to be able to achieve an optimal trade-off between the two. Choice of fonts entails a similar set of trade-offs, this time between screen real estate and readability. People who are willing to use a smaller font can fit more information on the screen. However, not everyone can read the smaller font sizes.

Preferred styles of working also affect choices of screen layout. For example, some people, such as E7 and E10 are very concerned about having a neat, well-organized work environment. E10 says, "I like things out of the way. I don’t like clutter on the screen. If there are too many windows, I get lost." E7 has a similar attitude. " I’m a clean freak -- I don’t like having extra stuff." In contrast, C3 likes to fit as much as possible on the screen. "You always need another little window." C6 likes to have "lots of X gadgets on the side of the screen" and C7 likes to "multiplex the screen" by using 8-10 windows on top of each other, with numerous icons in the empty corners of the screen. E13 has used the same screen layout for over two and a half years. Although he customizes often, he hasn’t bothered to remove items that he now longer uses, such as an Xload (which provides information about the current system activity) that he got from C6 a long time ago.

Other customizations are purely cosmetic and designed to increase the pleasure of using the system or to show off. These features generally appear as unusual screensavers[21] or backgrounds. The normal screensaver is somewhat boring and displays the letter "X" in random sizes in different places on the screen. Some people have spent many hours creating screensavers that perform various clever animations. One screensaver has a set of fish that move back and forth across the screen, disappearing if one fish intercepts another. Others show time, as clock with time elapsed since the last keystroke or interpreted as the current phase of the moon. Others show popular cartoon characters, pinups, and even a poem. D8 created an elaborate screensaver called "Kilroy was here". "Kilroy" peers up over the top of a random window on the screen, looks around, and then slowly disappears. D8 also created a special screensaver for Halloween, in which a witch appeared in one corner of the screen and "flew" across. He set up a workstation cluster so that the witch appeared to fly from workstation to workstation around the room.

Approximately half of the people also use a different background than the default gray. The simplest technique is to display an image chosen from an existing collection of images, e.g. a lion (B2), a castle (B5), or the moon (E2, A8). Other people devote additional time and create their own unique full-screen images (E10, D7 and D1’s image of his girlfriend). A number of people find background pictures distracting and choose either a uniform pattern, such as a grid of dots (E13, C2) or lines (C6). Others create their own random background pattern and repeat it across the screen (E12, D5, D6). D5’s background was copied by five people. It consisted of a variation of M.C. Escher’s dark and light birds in a subtle color combination that made it interesting but not distracting. Finally, some people like to customize just to be different. As E8 says, "I customize to be a little strange."

[21] A screensaver is an image that appears on screen when the system has not been used for a period of time. The screen saver literally "saves the screen", by not allowing any one set of pixels to be on all the time.

5.5.3 Seeking information about customization

Users were shown a set of information sources and asked which they used to help them customize their software. The information sources included: Athena's printed documentation, general-purpose books on Unix, "man" pages (Unix's on-line manual), an Athena-specific on-line help system, the on-line consulting system, asking someone, copying a file and experimenting with it, writing the customization files from scratch, and reading the source code to determine how to write the files. Participants rated each of these on a four-point scale: 1 = "never use", 2= "use rarely", 3 = "use sometimes" and 4 ="use often". Table 5-2 shows these responses, organized by job categories: A = manager, B = secretary, C = systems programmer, D = applications programmer, and E = other staff (non-programmers).

Table 5-2: Sources of information about customization

N	Job	Ask a person	Copy & exper.	Manual pages	Athena doc's	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
12	Sys. Program	1.8	2.1	2.4	1.3	2.3	2.0
8	App. Program.	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			

Columns are ordered according to the highest overall mean score, with the highest mean on the left. Group means are listed below each for each job category. The most common category is "ask another person", immediately followed by "copy and experiment" and reading the on-line manual (man pages). Reading the Athena documentation is less common and the remaining alternatives are used rarely or never.

The system programmers do not fit this pattern. They are most likely to read the manual pages or the source code or experiment with a copy of someone's file. The four most senior systems programmers (C9-C12) customize very little, and when they do, they are willing to ask for help. The rest of the systems programmers avoid asking for help from other people (mean = 2.25). Note the applications programmers are more like the rest of the organization. The rest of the staff like to ask someone, immediately followed by copying another person's files and experimenting with it. Reading the on-line manual pages is also seen as useful to everyone except the secretaries, who are more comfortable with the printed documentation. Because many of the people who answer questions for the on-line consulting system are available to the staff members, most people do not need to ask questions this way. However, some people use it extensively.

5.5.4 Encoding repeated behavior patterns

Customization is more than just choosing among a set of features. Most people reported that the most important kinds of customizations are those that allow them to encode commonly-repeated processes of behavior. When they discover they have been repeating a particular usage pattern over and over again, they devise a way to make the process happen automatically. Examples include: aliases (shorthands) for moving from one place to another and setting a number of options, laying out the screen (people often do this by hand for a while and then codify it).

E13 says, "I find I type the same thing over and over...if I find I do something regularly, then I automate it...It's not that I'm lazy, it's just that the computer can do it." E16 says, "If I do something a lot or execute the same commands in a row, I make an alias." A2 says, "Once the behavior is automated, I can do it without conscious thought so I can think about other things as well." E7 says, "You notice what you're doing all the time...you realize what you repeat...For example, I noticed that I always set up two Xterms and always stretch one. I always add a scroll bar." So she added them to her startup file. Note the similarity here between customizing the process of handling mail, via Lens, and the process of using a set of windows, through customizing the X window system.

The majority of customization occurs when people are faced with a new software application, especially if this occurs at the same time the user is starting a new job. Over time, most users drop to much lower customization levels. The most common form of customization in this later state is to encode commonly-repeated patterns of behavior (see Section 5.5.4 in Chapter 5). As A4 says: "Anything I do too much, I let the computer do for me." Or somewhat less enthusiastically from A9: "I reach an irritability level and then I bite the bullet and put it in my startup file." Users have several options in this environment. Probably the most common is the use of aliases, in which a word is equated to one or more commands, with the desired set of parameters. The line editor makes previously-typed commands easily accessible, but aliases provide a quick short-hand for commonly repeated behavior. Individuals make very different sets of decisions about when to create an alias,

when to scroll through and edit previously typed commands and when simply retype it from scratch.
.

This use of aliases allows users to encode patterns of behavior, which is almost always cited as more important that setting particular features of the workstation. The exceptions include items that directly affect the user's ability to work, such as font size or using reverse-video windows (for people who get headaches with the extra flicker in a white window).

5.5.5 Innovations

When asked to come up with an example of a customization innovation, many people felt that they had not been innovative. The most technical people did not consider their innovations to be customizations, since they were more likely to be actual programs. However, some people did describe interesting things they created. Some were very simple. E8 invented an emacs command that lets him automatically insert his signature. He has also redefined the TAB key so that file name completion works the same for Unix commands as it does within the emacs editor. The first customization allows him to automate something that he does repeatedly and the second allows him to maintain standard functionality across different software applications. D1's video image of his girlfriend and D8's innovative screen savers are both examples of innovations "for the fun of it".

A1, a technical manager, and C5, a systems programmer, each came up with the same innovation: they use the history file and the set lineedit command to create an editable menu of previously-typed commands. A1 and C5 each type many complex commands at the Unix shell and prefer this method to creating aliases. The disadvantage of aliases is that they are permanent and have to be explicitly deleted when they are no longer needed. They require a name, which has to be recalled, and do not make it easy to make minor variations in the command. In contrast, this use of the history file allows them to find rarely used commands and use them as models for new ones. After they are no longer needed, they eventually go away, as new commands replace the old.

Users in this study created several different mechanisms for organizing collections of tasks and using them under different circumstances. The most common method is to create aliases that perform collections of functions or display groups of windows and applications on the screen. Many users also create several initialization sequences and run them according to changes in their local work context. For example, A9 says: "I have to switch between this and a dialup environment -- so I have to put diddlely things in my .startup.tty file." Examples of the kinds of contexts include: the 'normal' setup used when using the workstation in the user's office , a limited setup for when the user was fixing equipment or addressing other problems 'in the field" (A9, E2, E3, E13), a programming setup to facilitate debugging (D1, C1, C8), and a "demo" setup for projecting an application on a large screen for visitors (A3, E1, E7).

D8 created another mechanism, which was widely copied, that allows users move the cursor off the screen and instantly access a new part of the off-screen memory, making it possible to interact with a different collection of windows running different applications. D1 found this a useful way to create two different working environments, one for programming (with a white background to keep him awake) and one for communication activities (with a more soothing black background). Systems such as *Rooms* (Card, 1987), are designed to explicitly support this kind of user task switching. These data support Card’s and Henderson’s observations that users organize work into collections of tasks. They also suggest that users will appropriate technology to organize context-specific collections of activities, especially if the system does not require them to articulate what the context is. Just as in the Lens study, in which users created different sets of rules and ran them when the context was appropriate, the users at Athena created different collections of applications and used them when the context was appropriate. The computer did not need to ’detect’ the change in the user’s context because it remained under the user’s control.

The most extensive innovations relate to Zephyr, the program that allows people to pop up messages on the screen. C1 developed a practical innovation: a "snooze alarm" made from Zephyr. C1 spends long hours in his office and sometimes wants to take a nap. He realized that that he could create an electronic alarm clock by sending a rapid sequence of 500 messages to himself (each with its own beep) at a specified time. He did this entirely within the customization features provided with Zephyr, but created a new, unanticipated use of the system as well. The customization features were intended to draw a user’s attention to a single new message. By executing them in rapid sequence, the result was noisy enough to wake him from a sound sleep. He says the design has the practical effect of a snooze alarm, because, "I can’t remember how to turn it off when I wake up. But by the time I remember how to turn it off, I’m awake."

Several other people expressed the desire to have an electronic reminder system. A2 wants a tickler system to reminder her of important events: "I would customize much more if I had time to, e.g. setting up Zephyr reminders.". D2 want an automatic warning when his Kerberos[22] tickets are about to expire. "I want to circumvent [the system], so when I [log in], it sets a [timer] for five hours later. At that time, it should pop up a window and ask me to [log in again]. A useful customization

[22]Kerberos allows users to identify themselves to the system and gives them permission to use applications for a particular length of time, usually eight hours.

feature would allow users to create general purpose timers which can run specified functions when triggered by different events, either a particular time or other kinds of events. This is not a particularly new idea and is provided in any 'event-driven' programming system, such as Hypercard or X Windows. However, in this environment, the technical skills required to do this effectively remain too high for most users.

One of the biggest problems with Zephyr is that a very large number of messages may appear on the screen in a short period of time. To remove a message, the user must click the mouse on the message. This works well when reading a few messages that are all important. However, some people have a large number of subscriptions (similar to electronic mail distribution lists). Zephyr users disliked the fact that all these messages appeared in one place on the screen. So the developer of Zephyr added a feature that allows people to have messages pop up on different parts of the screen. As people began to get overloaded with too many messages, they started dividing the messages into different categories. Just as in the Lens study, many users discovered that isolating personally-addressed messages and keeping them separate from more general broadcast messages is very useful. C7 extended this further, placing broadcast messages into a terminal window and allowing them to scroll by. Thus, he would have to look explicitly at personally-addressed messages and only look at the most recent set of broadcast messages.

The original version of Zephyr placed all messages in the upper right-hand corner of the screen. This was inconvenient for a number of reasons, and the developers added a customization feature that allows users to specify where to place messages. Users could also identify the messages by their source and place them on different locations on the screen. Users with large numbers of messages from general-purpose lists such as 'Help' had been finding themselves overwhelmed with messages and unable to find the important ones.

E15 described an innovation in how he customized Zephyr: "I wanted to direct messages to different parts of the screen. I monitor many [Zephyr lists]. I get two kinds of messages: things I want to read and things to look at if I'm already looking on the screen. I run two windowgram clients and put one in an Xterm. The junk messages are there, and they just scroll away. This way, I can segregate based on importance." Basically, the important messages pile up in a stack on his screen and he has to click on them to make them go away. The "junk" messages appear in a window and scroll away automatically. So the window always contains just the most recent messages and the others disappear. He says "It's VERY useful -- otherwise I couldn't get the right level of information." He said that this was "a painful enough distinction to do that customization." He did this before the new version of Zephyr and hasn't upgraded it to meet the new versions customization process. However, Zephyr has been modified so that users can make this kind of customization. Interestingly, this matches a highly-valued type of rule created in the Lens study. Mackay et al. (1989a) reported that 85% of the Lens users also found it useful to distinguish personally-addressed mail messages from those to distribution lists.

Zephyr was created to imitate the kinds of messages that system managers of time-sharing systems send out when the system is about to go down or when file servers are getting full. Because Athena is distributed rather than centralized and because users are not tied to any particular hardware, the standard Unix version did not work. Zephyr has dynamically-created ’subscription’ lists. Whenever a user logs into a particular workstation, he or she is automatically subscribed to information sources about the local printer, file servers, etc. If any of them go down while the user is logged in, the user receives a Zephyr windowgram. Users quickly figured out that people could use this as a way of finding out who was located where around the campus. Shortly thereafter, a feature was added to allow people to "hide" from such inquiries by others on the system. The subscription lists were extended to be more like electronic mail distribution lists, in which people could add themselves to the list. The lists tend to have a very transitory sense to them, because they deal with transitory information. (The user only receives these messages if he or she is currently logged into the system.) An example of a list is one for people who like to go out for late-night pizza or other food. Sending such messages over electronic mail would be annoying, because the messages would persist long after the appointed time for the pizza. With Zephyr, only the people currently logged in and subscribed to the list receive the messages -- which is appropriate.

A number of students were frustrated that they could not use Athena’s official on-line consulting (OLC) system late at night. So they created their own ’help’ list which anyone can subscribe to. People with questions can send them to the Help list and other students who may know the answer send back answers. Only people who are currently logged in see the messages and are thus not bothered with messages that are now answered. Some questions provide a flurry of responses while others are more difficult and may be referred to OLC. Essentially, the students have developed an informal, self-regulating peer-peer, consulting system, which operates as a preliminary filter for the more formal on-line consulting system. Many questions are answered before they ever get to OLC. The Athena consulting staff now monitors the help instance and often provide answers themselves. They encourage this activity because it helps to reduce the pressure on their already overloaded system. Thus the students appropriated Zephyr, a message system originally designed to handle system messages, and created their own peer-to-peer consulting system. This has affected the consulting staff, who have shifted their priorities and run the consulting operation differently as a result.

The changes in the use of Zephyr affected the future development of the on-line consulting system and the operation of the consulting group. The consulting organization might have viewed this activity as a threat. Instead, they investigated how it could be used to help them reduce the load on the existing consulting service. According to their manager, A2: "The consultants are customizing OLC by writing alternative clients that sit on top of the regular one -- the clients that "Athena central" has coded are automatically parasitized by the "competition": the consultants have coded an emacs interface written on top of the Athena Central one. We just updated the central and this automatically updated the parasites."

User’s expectations in the informal help system are different from those with the official consulting service. In the former, users know that they’ll only get an answer if someone who knows the answer happens to be logged in at the time and is interested in answering the question. Thus their expectations that it will be answered are relatively low. On the other hand, questions to OLC go into a queue and the user expects the question to be answered within a reasonable period of time. Users thus have a quick, low overhead method to try first and a more formal method to try if it doesn’t work. The consulting group can prevent questions from clogging up the official queue by encouraging questions to be answered quickly though the Help system -- or answering those questions themselves. Since the help system is informal, questions aren’t logged and the same question can be answered slightly more quickly than in OLC. The extremely technical consulting staff can continue to monitor the formal channel, which tends to get the more difficult questions, and the newer or less experienced consulting staff can both learn from and help to answer questions in the informal channel. A2 is currently considering a reorganization of consulting and would like to more actively support this kind of peer-to-peer consulting.

5.5.6 Maintaining stability in the face of change

Although one purpose of customization is to allow people to express individual preferences, a much more common use of customization is to maintain stability in the face of change. Change takes many forms and people’s reactions to it vary. People customize as way to maintain stability when moving to a new software environment, when constantly switching between hardware and software platforms (e.g. using a Unix workstation at work and a personal computer at home), and when handling a software upgrade. In the Athena study, everyone who had customized their window manager software managed to maintain stability through one of several means. Some people refused to change window managers, others moved to an even newer window manager and retrofitted it, and everyone else retrofitted the new standard window manager to be like their customized version of the old standard window manager. Only the people who did not customize their window managers at all did not perform new customizations. Note that this meant that all of these people were thus forced to learn to use the new window manager.

Carryover from previous software experiences Many of the people in this study had used different computer systems before coming to Athena and had established preferred patterns of behavior. In many cases, these earlier systems established the user's idea of what a system "should be". Thus A10 feels that MS/DOS, which she learned first, "has more functionality" than Unix and she doesn't "need to customize [MS/DOS] as much because it has lots built in". She continues to use both systems, because MS/DOS has a superior spreadsheet, which she uses on her IBM pc at home. She feels that the defaults for MS/DOS are "right", although she says doesn't mind using different commands in the two systems, e.g. "clear" versus "clr". She clearly thinks in DOS, not Unix, terms. Similarly, E13 claims that "'lo' isn't the natural way to logout, 'kk' is."[23] C2 also used TOPS-20 before coming to Athena. "I have some dot files that date back to 1985. I also have some abbreviations from TOPS-20 (a DEC operating system) that I make work here." E15 used Multics before coming to Athena in 1983. "I copied over ideas and things from Multics and retrofitted things to make Unix emacs like Multics emacs." When D5 arrived, he was used to the VMS operating system and created an alias, 'dir', for the directory command. He now uses 'ls', which is the Unix version of the command, but hasn't removed the old alias from his files.

Although they don't involve customization directly, early experiences with a text editor may also affect users for a long time. C9, C12 and D3 continue to use the *vi* editor, rather than Athena's standard editor, *emacs*. D3 says that "emacs isn't very good. I want auto-line-wrap and it's unusually difficult...whereas vi does it for you just fine." This choice affects their use of other software. They avoid using certain features that assume emacs key bindings, such as 'line edit'. Most other users find this to be a very useful way to edit rather than retype previous Unix commands.

Users who adopt other people's customizations must often deal with the consequences of that decision for years to come. Thus, E1 and others borrowed D6's customization files and continues to use them years later, despite other changes in the system. Individual choices about customization also affect future behavior. Thus, users who create aliases so that commands in Unix look like commands in the previous system may reinforce their reliance on the previous system and fail to learn new aspects of the new system.

---

[23]'kk' is the command used to logout from the TOPS-20 operating system which he learned before learning Unix.

Past experiences may have a big impact, particularly past failures. Several people said that if they had a bad experience with a piece of software, they would be unlikely to try it again. For example, D3 says: "I spent two days on Zephyr and it never worked. So my Zephyr subscriptions don't work. Once browned off, I don't come back unless it's a real pain." E15 and D7 each provided similar examples in which they had tried something, given up, and not tried newer versions. This same effect was seen in the Lens study, in which members of the original group were reluctant to try subsequent versions of Lens and several never tried it again.

Maintaining stability across platforms

Another kind of stability is important for people who work regularly on more than one workstation or have a personal computer at home. According to C3: "I want the screen layout to look OK on every platform, so I make it adjust automatically." D7 has a different strategy. "Now I'm very conservative. I use lots of different platforms...so I'm trying to get myself used to the things that cross systems." Rather than customizing the software, he 'customizes' his own behavior. He says, "It's easier for me to learn than to take time to customize." D8 tried to accommodate switching between two different keyboards by rearranging the keybindings. However, this didn't work as well as he expected. "Now I make mistakes -- this hurts me. My fingers used to do the right thing on both keyboards."

Reacting to a system upgrade

One of the most common instigators of change in this environment is a system upgrade. People react in different ways. Some simply let the system break down over time and work with successively less functional versions of the software, like A2. Others resist making major investments in learning transitory software, such as C12. "I used to customize more, but then I'd lose it. The update procedure was agonizing -- I'd have to spend a day fixing it. Replacing software is smoother now, but I'm so fried, I'm not adventurous any more." C9 says, "I gave up on this shit five years ago -- it's too hard to keep up." However, most people use the system upgrade to reflect on their use of the software and customize to maintain certain desired patterns of behavior. As E2 says, "I just try to maintain equilibrium every time the system changes...I don't want to change -- I don't have time. I'm pretty busy." A few also use it as an opportunity to explore new functionality that may have been provided in the update, like E16. "I ported my existing Zephyr customizations from the old system to the new, and now I put window grams in different places on the screen."

D2 retrofits each new window manager because he says: "I don't want to relearn the key bindings." E1 does the same thing. Although she feels that her job compels her to use a "vanilla" workstation environment to keep compatible with mainstream Athena users, she draws the line at her window manager key bindings. She refuses to change them. "I got a set of custom bindings from D6 about four years ago. Now they're ingrained...I have a set of key/mouse combinations for my window manager functions. It's just the way I do it -- I'm too lazy to change." On the other hand, this costs her sometimes. "I go crazy when I use someone else's workstation."

Table 5-3 summarizes how people reacted to the change in the default window manager, from UWM to MWM, in the latest system software release. The table omits 11 individuals who have been at Athena for less than six months and have only used one window manager (MWM) in that time.

The first four columns in Table 5-3 list the person's individual code, job type, technical level, and years at athena. The technical level (high, medium or low) is derived from a combination of the user's computer education and previous job experience, if any, as a programmer. Within these subgroups, individuals are ordered by years at Athena.

Column five indicates whether or not the person chose to switch from the UWM window manager to the MWM window manager. (Note: the members of the video group deferred this decision until D3 figured out how to retrofit all of their customization files to handle MWM. So they are listed as having switched, but did not do so immediately.) All but 11 people (73%) switched to MWM. Of those who did not switch to MWM, five had already switched to another window manager, called TWM (Column six). TWM is available only with the beta test version of the upcoming system software release and is still in the testing phase. Six people (15%) refused to switch to MWM at all.

Column seven shows which people had already customized the UWM window manager. Nine people (23%) did not customize UWM at all while the rest (77%) did. Column six shows the people who switched to MWM (which became the new standard window manager) and column seven shows those who switched to TWM instead. Column eight shows the people who retrofitted the new window manager to be like the old. (Notice that this column does not apply to the six people who refused to leave the original UWM window manager. It also does not apply to people who did not customize the original window manager.) What is striking about column eight is everyone who had previously customized their window manager and switched to a new one, retrofitted the new window manager. Column nine identifies the same phenomenon in a slightly different way. It shows that 78% of the people maintained a constant environment, either by refusing to switch (6 people or 15%) or by retrofitting their existing set of customizations to make the new window manager look like the old one (25 or 50%).

Table 5-3: Reactions to changes in the window manager

Code	Job Type	Tech. Level	Years Here	Customized UWM?	Convert to MWM?	Use TWM?	Retro fit?	Stay same?
A1	MG	High	3	Y	N	Y	Y	Y
A5	MG	High	3	N	Y		--	N
A3	MG	Med	3.5	Y	Y		Y	Y
A4	MG	Med	1	Y	Y		Y	Y
A2	MG	Low	6	N	Y		--	N
A9	MG	Low	4	N	Y		--	N
A8	MG	Low	3	Y	Y		Y	Y
A7	MG	Low	1.5	Y	Y		Y	Y
B1	AS	Low	2	N	Y		--	N
C2	SP	High	4.5	Y	Y		Y	Y
C3	SP	High	4	Y	N	Y	Y	Y
C4	SP	High	4	Y	N	Y	Y	Y
C5	SP	High	3.5	Y	Y		Y	Y
C11	SP	High	3.5	N	Y		--	N
C6	SP	High	3	Y	N			Y
C9	SP	High	3	Y	Y		Y	Y
C12	SP	High	2.5	N	Y		--	N
C7	SP	High	2	Y	N			Y
C1	SP	High	1.5	Y	Y		Y	Y
C8	SP	High	0.5	Y	N	Y		Y
D7	AP	High	4	Y	N		--	Y
D5	AP	High	1.5	Y	Y		Y	Y
D6	AP	Med	5	Y	Y		Y	Y
D2	AP	Med	3	Y	Y		Y	Y
D4	AP	Med	2	Y	Y		Y	Y
D1	AP	Low	6	N	Y		--	N
D3	AP	Low	1	Y	Y		Y	Y
E15	OS	High	2.5	Y	N		Y	Y
E1	OS	Med	4	Y	N		--	Y
E3	OS	Med	4	Y	Y		Y	Y
E2	OS	Med	3.5	Y	Y		Y	Y
E16	OS	Med	2.5	Y	Y		Y	Y
E6	OS	Low	5	N	Y		--	N
E4	OS	Low	2.5	Y	Y		Y	Y
E13	OS	Low	2.5	Y	N	Y	Y	Y
E7	OS	Low	2	Y	Y		Y	Y
E11	OS	Low	1.5	N	Y		--	N
E14	OS	Low	1	Y	Y		Y	Y
E5	OS	Low	0.8	Y	Y		Y	Y
E8	OS	Low	0.8	Y	N			Y

N = 40
20 (50%) Stayed same: changed to MWM and retrofitted the software.
5 (13%) Stayed same: changed to TWM and retrofitted the software.
6 (15%) Stayed same: refused to change window managers.
9 (23%) Adapted: changed to MWM with no customizations.

What is particularly interesting is the relationship between initial investments in customizations and the reaction to change. Everyone who customizes software (Column five) chooses a strategy that allows him or her to maintain a constant environment (Column nine). Similarly, everyone who does not customize the first window manager also accepts the automatic conversion to the new window manager and adapts his or her behavior accordingly. Thus, in this group, previous decisions about customization correlate perfectly with a desire to maintain stability in the face of change or a willingness to accept change.

Why did the nine people not customize in the first place and not react to the change in the new window manager by adding customizations? Three people (A2, B1, E6) simply did not have the knowledge to make any customizations, nor do they have other people to help them adjust to the new environment. They were thus forced to learn the new window manager. Five people (A5, C11, C12, D1, E11) refused to customize for philosophical reasons, prefering to pay the costs of converting rather than creating customizations that would always be out of sync with everyone else. One person (A9) simply felt that it is easier to learn the new window manager than retrofit it to be like the old.

Although all of the people who had invested in customizing their window managers and converted to a new window manager invested additional time in retrofitting it, there is evidence that too great a change will cause people to abandon their customizations entirely. In the case of Zephyr, the method of customizing changed drastically in this release. D7 described his experience, after he had spent quite a bit of time customizing the earlier version of Zephyr. "Then they changed everything. So I gave up. I’m a conservative guy -- there has to be a compelling reason for me to go back over that threshold." It’s also important to note that once a person has tried something and been burned by it, he or she is often reluctant to try it again.

The people who do not customize have a variety of reasons. Some have made an active decision to use the standard commands, whatever they are. (e.g. E10 says, "I’m trying to force myself to learn the standard commands.") Note that there’s an assumption that the commands that are given as Athena defaults are in some way "standard". Several people who move from one Unix system to another pointed out that a number of Athena’s commands are not standard (C9, C7, C12). C7 says, "I back out of lots of Athena ’standard’ customizations because I’ve been a Unix user for over eight years. If I go to other sites, I want standard Unix." Many of these are based on the preferences of some individual many years ago and they have found their way into the standard files. The most experienced systems programmers want to use standard commands that will be the same across Unix platforms.

Maintaining customization files

Customization files require maintenance, not only because the underlying technology changes, but also because the individual's job and people they contact change. For example, many users create "alias" files to keep track of people they contact often. One alias might equate a friend's nickname with his or her network address. Another might use a special program to find out whether or not a friend is currently logged in. As people move away, group boundaries change, and job requirements change, some of these aliases will no longer be used. A few people are very careful to maintain their files, deleting items as they cease to be relevant. However most do not. For example, in this study, only one third of the people who had aliases kept them up to date. Only one person (D1) reported cleaning up his files on a regular basis. Those who do maintain their files do so when they were looking in the file for some other reason, typically when adding a new alias. Very few people were aware of how many of their aliases they used, or what percentage of their other customization files continued to be relevant.

This leads to a very important point. Customization files become obsolete. Sometimes this has no impact except to take up unnecessary room on the system. Several people also commented that it serves as a historical record, allowing people to remember what they were doing at different points in their jobs. However, some customizations, particularly those that affect patterns of behavior, continue to affect the user. As they become less relevant, many users are unaware of the changes. They do not receive feedback about the effectiveness of their customizations or even the level of use of those customizations. Thus, they do not have effective methods of maintaining them.

Table 5-4 summarizes how the members of each group use and maintain aliases. Aliases provide an important mechanism for encoding repeated patterns of behavior. Examples include specifying a preferred list of options to a command, looking up a phone number from a file, setting up a particular arrangement of windows.

Aliases provide another mechanism for avoiding the learning costs associated with converting to new software. Because a significant investment in an existing system has already been made, people are reluctant to get rid of that investment and start over with the new system. However this presents problems, both in the short-term and in the long term. In the short-term, people may change the basic syntax of their interactions with the new system, but this hides the underlying "philosophy" behind it. As mentioned before, A10 perceives Unix as being less functional than MS/DOS, because she already understands DOS and is trying to simply use that knowledge rather than learning about the new, often more powerful, possibilities of Unix. Thus, the creation of

Table 5-4: Maintenance of alias files

Job category	Use 0-60%	Use 80-100%	No aliases	Total
Managers	4	2	4	10
Secretaries	0	0	5	5
Systems Programmers	5	3	4	12
Applications Programmers	6	1	1	8
Other staff	6	5	5	15
Totals: Percentages:	21 42%	11 22%	19 38%	51

customizations to make the new system look or act like the old may make the new system appear to be less functional. In the long run, these decisions may cause problems when the user has to work in other Unix environments (because the customizations won't be there, the user won't know how to use the basic functions). Some of this can be alleviated by improved user interfaces, which allow people better opportunities for discovering aspects of a particular system, but this won't solve the whole problem.

An advantage of this technique is that it allows the user to decide *when* to make conversion decisions. If a conversion occurs at a particularly busy time, the user can put off paying the costs of learning it until a more convenient time. The problem is if the person puts off the learning forever. In such cases, the user will continue to work at a suboptimal level, working only at the lowest common denominator between the two systems.

Chapter 6

Patterns of Sharing Customizable Software

Much of the research in computer-supported cooperative work develops or examines software that is explicitly designed to support information sharing, such as electronic mail, group-oriented decision tools, project management tools, and electronic calendars (Malone et al., 1987a, Borenstein and Thyberg, 1988, Stefik et al., 1987, Begeman et al., 1986, Sathi et al. , 1986, Greif and Sarin, 1986). However, other types of information sharing occur naturally within organizations, sometimes in unexpected ways. In this chapter, I discuss a phenomenon that occurs with user-customizable software: when members of organizations share their customization files, they effectively establish and maintain standard patterns of behavior throughout the organization. After providing evidence for this phenomenon, I discuss some of the implications for both organizational theory and system design.

The activity of customizing one's own software applications might be viewed as a reasonably solitary task. After all, the goal is often described as allowing each user to express his or her *individual* preferences, or as pointed out in Chapter 5, to allow individual users to maintain a consistent environment, despite changes the hardware, software or work environment. However, customizations are rarely independent; users may be greatly influenced by how other members of the organization customize their workstations. 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 the time spent learning how to customize, 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 through an organization or over time, rather like tracing the spread of archeological artifacts. 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, provides a view of a communication network within the organization that has on-going impact over time.

This chapter discusses the social aspects of an apparently non-social practice, that of customizing one’s software environment. I begin by presenting a network diagram of who shares customization files with whom in the organization. I examine additional factors such as membership in groups (both before and after a reorganization), office mates, changes in jobs and general communication patterns among staff members. I then explore customization sharing in more detail through four case studies of groups or individuals who share their customization files. I then examine the circumstances under which sharing of customization occurs, including: who shares customizations, when they are shared, the methods by which they are shared, and the factors that affect sharing. I conclude by suggesting implications for organizational theory and software design and discuss the implications for research in learning and motivation.

6.1 What are the patterns of sharing customizations?

Table 6-1 provides a summary of who shared customizations with whom in the context of a major staff reorganization and office change. The reorganization occurred in January, at the beginning of the study, and affected 85% of the staff. The office change occurred in March, during the second half of the study and affected almost half (37/80 or 46%) of the staff.

Column 1 indicates the code for each individual staff member who participated in the study. The total staff has remained between 75 and 85 people during the course of the study, but several people have left and others have arrived. Managers are designated with an "A", secretaries with a "B", systems programmers with a "C", applications programmers with a "D" and other non-programming staff with an "E". The latter include documentation writers, faculty consultants, operations staff, etc. Individuals who completed the first part of the study but have since left, and people who did not participate at all are not included in the table.

Column 2 in table 6-1 identifies the people with "open" files. In this environment, people have the option of making their directories accessible to others in the organization (known as making one’s directory "world-readable"). Most of the people who do this want to make their customization files (and programs) available to others who might want to copy working examples for their own purposes. Several public locations are also set aside for such files, but some people prefer *not* to post their files there, so that they do not have to support them.[24] 45% of the study participants keep their files open. Of these, 20% of the managers make their files accessible and none of the

[24]If a file is copied from a directory, there is no implied support by the author. However, if the file is given to others in a public forum, then the author also implicitly accepts a certain amount of responsibility for making sure that the file is correct, answering questions and maintaining it over time.

Table 6-1: Customization exchanges within the organization

Code	Files Open?	Changed Office?	Officemates: Pre	Shared Post	Reorganization: files?	Pre	Got files Post	from:		Gave files to:	
A1	N	N	C8	C8	Y	SD	AD	C8	+3	A6 C8	
A2	N	N	xx	xx		US	US	E16	+1		
A3	N	Y	E4	A7	N	US	US	E16		A7 A10 B1 B5	
A4	Y	Y	C5	C15	N	US	US	C3 E16	+1		
A5	N	Y	xx	xx		US	US				
A6	N	N	xx	xx		OP	SO	A1 D5 E11		A7 A10 E9	
A7	N	N	D1	A3	Y	OP	SO	A3 A6 E9 E13			
A8	N	Y	xx	xx		VCG	VCG	D1 D2 D3 D4 D5 E5 E7			
A9	Y	N	Ex	Ex	N	NET	NET	D8 E15	+1		
A10	N	N	xx	xx		ADM	ADM	A3 A6 D6 D8 E16	+1	B1 B2 B3 B4 B5	
B1	N	N	xx	xx		ADM	ADM	A10 A3 E2 E16	+4		
B2	N	N	xx	B4	Y	ADM	ADM	A10 E2		B4	
B3	N	--	--	xx		ADM	ADM	A10 B4 E2	+1	B5	
B4	N	--	--	B2	Y	ADM	ADM	A10 B2 E2		B3	
B5	N	--	--	xx		ADM	ADM	A10 A3 B3 E3	+1		
C1	Y	N	C13	C6	N	SD	SO	C4	+4	C5 D5 E14 E16	
C2	Y	Y	D4	C16	N	SD	AD		+6	C5 C7 C12 E4 E13 E16	
C3	Y	Y	xx	C4	Y	SD	SO	C4 E13	+4	A4 C4 C5 C10 D7 E4 E13	
C4	Y	Y	Cx	C3	Y	SD	SO	C3	+1	C1 C3 D7 E3	+2
C5	Y	Y	A4	Cx	N	SD	SO	A1 C1 C2 C3 C4	+1	E14	
C6	Y	Y	xx	C1	Y	SD	NET	C7 E16		E13	
C7	Y	Y	D9	D9	Y	SD	SO	C2 D7 D8 D9	+3	C6 D9	
C8	Y	N	A1	A1	Y	SD	VCG	A1		A1	
C9	N	N	Cx	C15	N	SD	AD	DG	+1	C10 E13	
C10	N	Y	--	Ex	N	--	SO	C3 C9			
C11	N	Y	Cx	C12	N	SD	SO	E3			
C12	N	N	xx	C11	N	SD	VCG	C2	+1		
D1	Y	Y	A7	xx		OP	EI	D8	+2	A8 E6	
D2	N	N	D5	D5	Y	VCG	VCG	D4 E1	+4	A8 D3 E5 E7	
D3	Y	Y	D10	E7	Y	VCG	VCG	D2 D4 D5 E5	+1	A8 D2 D4 E5 E7	+5
D4	N	Y	C2	xx	N	VCG	VCG	D3 D5 E5		A8 D2 D3 E5 E7	
D5	N	N	D2	D2	Y	VCG	VCG	C1	+1	A6 D4	
D6	Y	Y	xx	D7	N	US	EI	E1	+4	A10 E1	+4
D7	Y	N	xx	D6	N	US	EI	C3 C4	+4	C7 E2	
D8	Y	--	--	E16	Y	--	EI	E16		A9 A10 C7 D1 E16	
E1	Y	N	A13	Ex	Y	US	EI	D6 C1 E16	+5	A13 A14 D2 D6 E13	+n
E2	N	N	xx	xx		US	US	D7		B1 B2 B3 B4	+3
E3	Y	N	E19	E14	Y	SD	SO	C4		B5 C11	
E4	Y	Y	A3	Ex	Y	US	US	C2 C3 E16	+4		
E5	Y	Y	xx	xx		VCG	VCG	D2 D3 D4 D5 E7		A8 D3 D4 E7	
E6	N	N	xx	xx		OP	SO	D1			
E7	N	Y	xx	D3	Y	VCG	VCG	D2 D3 D4 D5 E5		A8 E5	
E8	Y	Y	Ex	Ex	N	US	US				
E9	N	Y	--	E13	Y	--	OS	A6 E13		A7	
E10	N	--	--	xx		--	OS	C8 E13	+2		
E11	N	N	Ex	E19	Y	US	US		+4	A6	
E12	N	--	--	xx		--	US		+2		
E13	Y	N	E14	E9	Y	OP	SO	C2 C3 C6 C9 E1	+2	A7 C3 E9 E10 E14	
E14	N	Y	E13	E3	Y	OP	SO	C1 C5 E3 E13			
E15	Y	N	Ax	Ax	Y	NET	NET	C1 C5	+8	A9	
E16	Y	Y	xx	D8	Y	US	EI	C1 C2 D8	+4	A2 A3 A4 A5 A10 B1 C7 D8	
	45%	51%			62%						

Key:	-- Not here	A = manager	SD = Systems Development	SO = Systems/Operations
	Cx Shared w/ C	B = secretary	US = User Services	AD = Advanced Development
	xx Not shared	C = sys. program	OP = Operations	EI = Educational Initiatives
		D = app. program	VCG = Video group	
		E = other staff	NET = Telecommunications	+n = exchanges with
			ADM = Administration	people not in the study

secretaries do. However, 75% of the systems programmers, over 50% of the applications programmers and 50% of the other staff make their files accessible.

Over half (51%) of the study participants changed offices. Columns 3-6 indicate whether or not the particular individual changed offices, who the officemates were both before and after the office change, and whether or not the office mates exchanged files. "--" indicates that the person was not here at the time, 'xx" indicates that the person had an individual office, and a capital letter followed by an x (e.g. Ex) indicates that a staff member who did not participate in the study shared the office. (In this example, the E indicates that the person was a non-programmer.) Note that only 62% of the office mates exchanged customization files.

Columns 7 and 8 identify the group to which the staff member belonged both prior to and after the reorganization. These groups are also shown graphically in Figures 6-1 and Figures 6-2. Column 9 identifies the people who the staff member got customization files from and column 10 identifies who the staff member gave files to. The number to the right of each column indicates the customization file exchanges with people who did not participate in the study. The original author of the window system is cited four times and two others are cited five and seven times. The rest are cited once or twice.

Figures 6-1 and 6-2 show how the staff were organized prior to and after the reorganization. Two groups, the administrative staff and the video group, were affected primarily through the addition of personnel. The Telecommunications group has a matrix relationship with another MIT organization, Information Systems. This group also gained members as a result of the reorganization. Members of the Systems Development were distributed to different groups and a new group called Systems and Operations was created. The remaining part of Systems Development became Advanced Development. The User Services group previously consisted of four subgroups. Two functions that were formerly performed by one or two individuals (the information officer and the two faculty liaisons) have been turned into two new groups (External Relations and Educational Initiatives). The consulting, documentation, and training functions continue to be part of User Services, under a new manger.

Figure 6-3 displays the customization exchange patterns in the organzation prior to the reorganization. The exchange of customization files is indicated by arrows. A one-sided arrow indicates that one person received one or more files from another. An arrow may indicate exchanges in which the author is proactive, as when E2 gave files to B1. It may also indicate exchanges in which the recipient is proactive, as when C8 looks for examples in A1's files. Two-

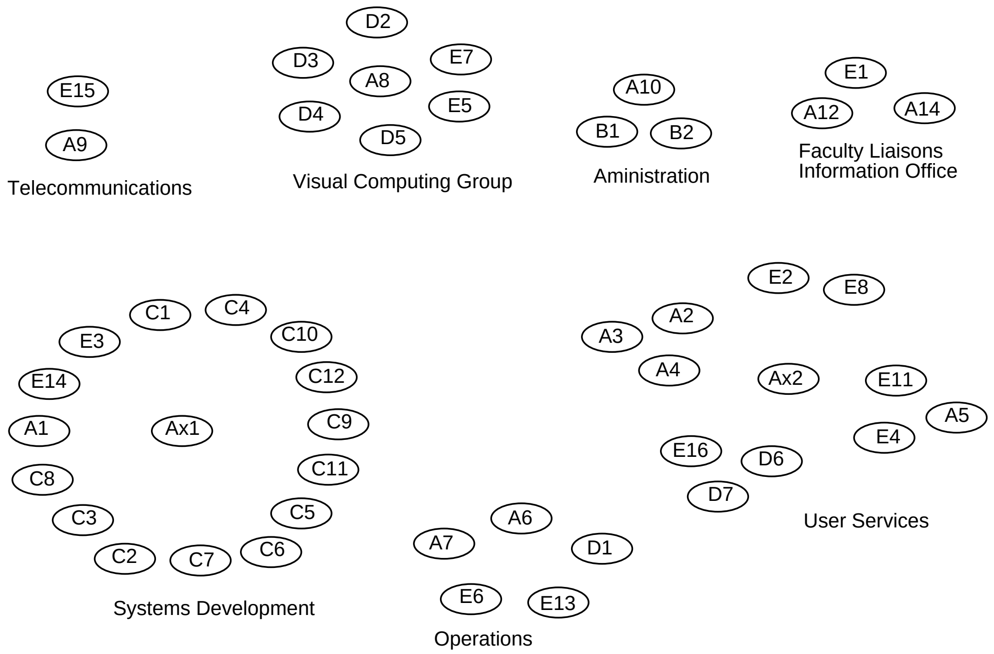


Figure 6-1: Groups prior to the reorganization

sided arrows indicate an exchange of files. Note that this graph does not indicate how many exchanges occur over time, it simply shows that at least one exchange occurred. Several of these exchanges involve a single file that is popular and copied by several people. For example, D5 has a background pattern that many people have copied and D8 has a technique for creating multiple collections of windows on the screen. In each case, the person has given only that item to other people.

The most striking aspect of the figure is the relative isolation of the video group. They actively exchange customization files among themselves, but exchange very little with people outside the group. The adminstration group receives a number of files, particularly from E2 and A3, and A10 provides files to the other members of the group. The members of the remaining three groups appear to exchange files across groups as well as within groups. Note E16, who receives files from the systems development group and provides them to members of other groups.

Figure 6-4 shows the exchange of customization files after the reorganization. Note that the video group is still isolated, although they have gained a few members. The administrative group still

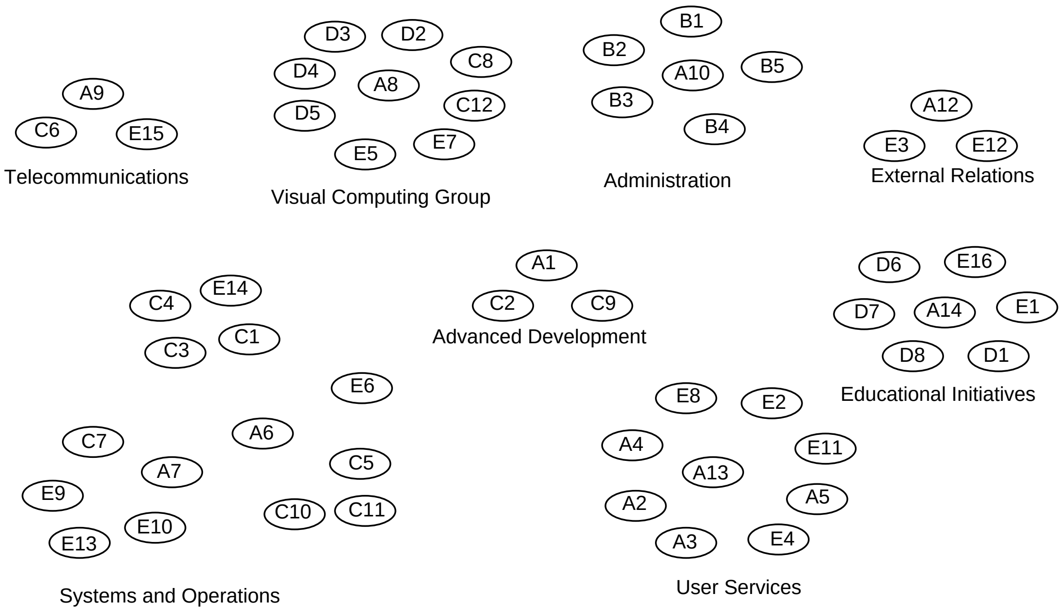


Figure 6-2: Groups after the reorganization

receives more files than they give and A10 actively receives files from a number of sources and provides them to other members of her group. The User Services group appear to share less with each other than with members of other groups, and there is a significant amount of sharing between the Systems and Operations group and the Educational Initiatives group.

Figure 6-5 shows how the staff are located geographically (after the office change) and indicate which officemates exchange files with each other. Officemates are indicated by two ovals that touch each other. Empty ovals represent staff members who did not participate in the study. The two arrows that extend beyond the office pairs indicate sharing that occurred between former officemates.

Of the participants in the study who shared offices, 23/37 (62%) shared customization files with their officemates. In the two reorganizations, there were 35 distinct office pairs.[25] Of these, 17/35 (49%)

[25]Some people kept the same officemates during the office change and others changed. The first statistic refers to the percent of the people in the study who shared offices at any time. The remaining statistics count some people twice, if they had different officemates in the two different office arrangements.

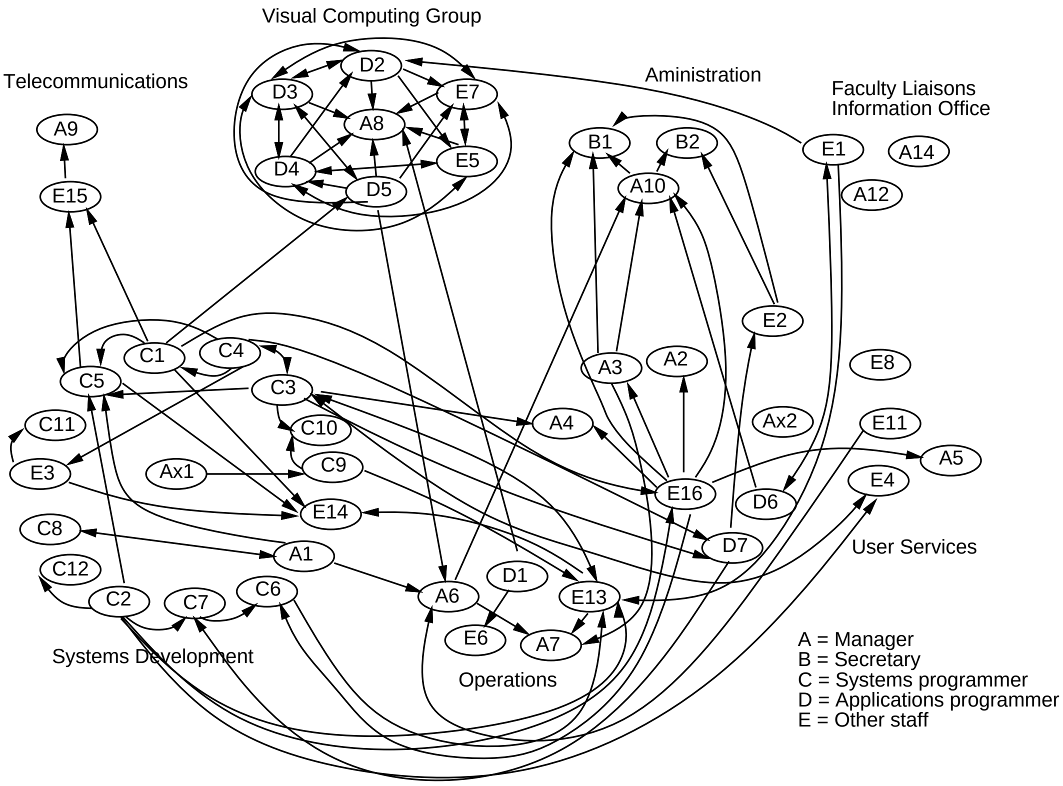


Figure 6-3: Pre-reorganization: Network of customization sharing

shared files with their officemates. Of those who shared files, 4/17 (24%) were equal exchanges in which both people gave and received files and 13/17 (76%) involved only one person giving files to the other.

6.2 Case studies of sharing customizations

The following four sections describe examples of sharing of customizations within this organization. When describing the process of sharing, people often slip into a description of sharing general knowledge about the system and sharing innovations.

6.2.1 Sharing within a cohesive group

One Athena group works on the design and development of multi-media applications for Project Athena. Some members of the group are working on the underlying system issues involved in adding video to the X Window system. Others are developing an authoring language for multi-media education software applications. Still others use this language to create courses and

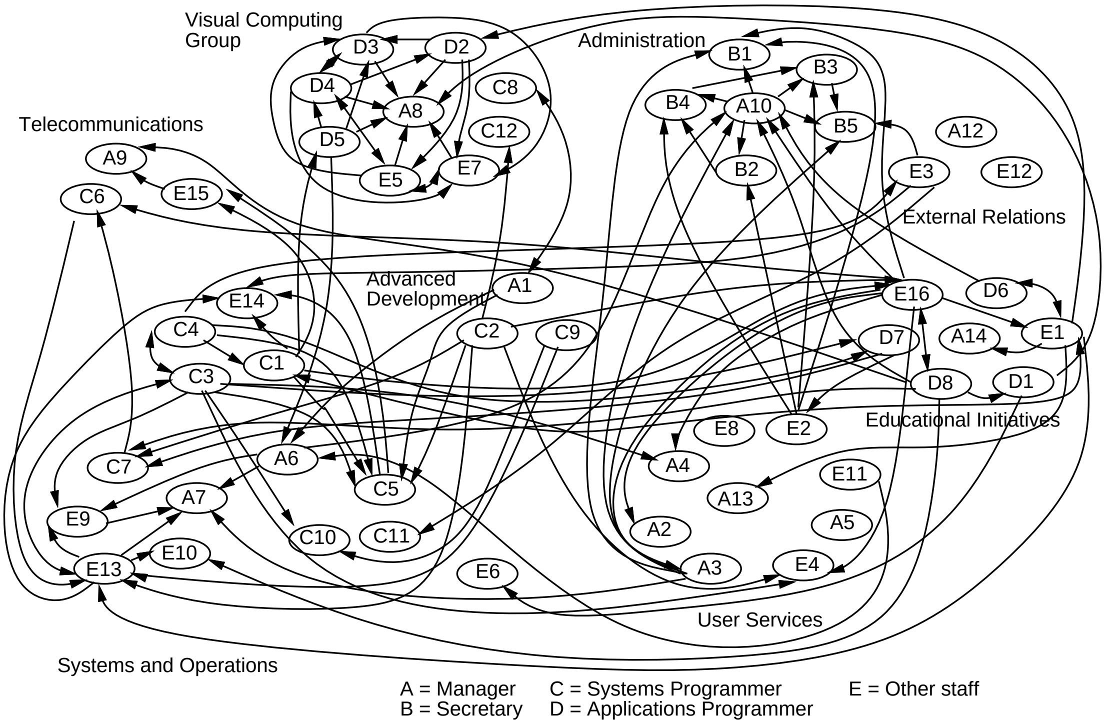


Figure 6-4: Post-reorganization: Network of customization sharing

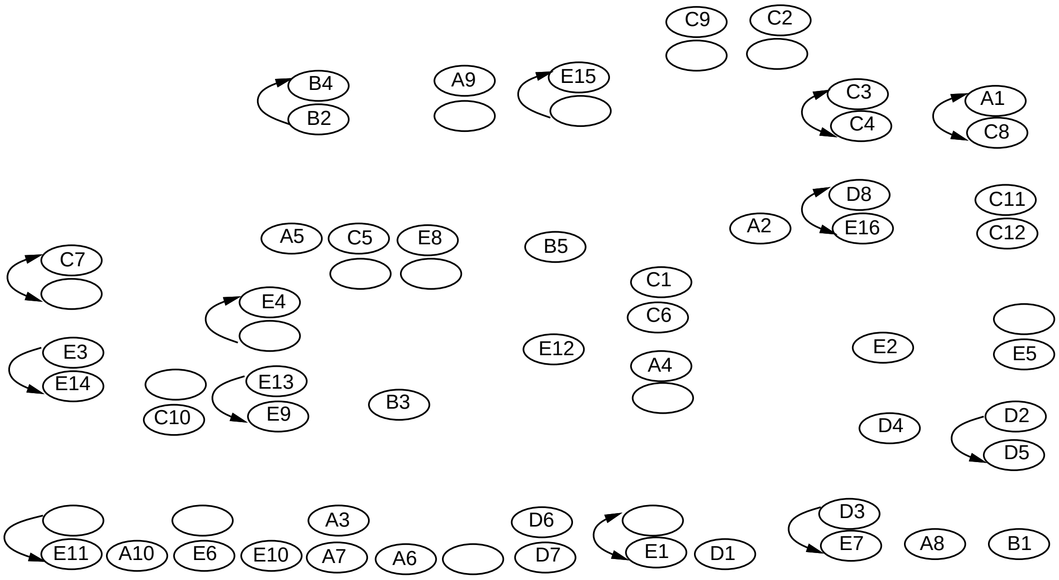


Figure 6-5: Geographic location: Network of customization sharing

demonstrations. This group is a small group with a manager and seven staff members (A8,C12,D2,D3,D4,D5,E5,E7). Students and interns periodically join the group for particular projects. They all use color workstations that can accommodate video inputs, which makes their hardware somewhat different than that of other members of the Athena staff. The group is very cohesive, partly because they have been fighting for a cause they believe in. They are friends as well as colleagues and share an excitement about their work.

In the early days of the group, approximately three years ago, one person (D4) took on the role of helping the other members of the group customize their workstations. He sent out electronic mail messages, particularly after a system upgrade, describing how to modify the software to handle the change. He personally helped others set up their workstations. Although one of the more technical members of the group, he is not the most technical. However, he felt strongly that the group needed help, particularly because they had unique requirements and were differentially affected by changes in the hardware and software environment. For example, when faced with such a change, he would figure out what to do and then pass the changes on to the other members of the group. D4 thinks carefully about the needs of the individuals in the group and only offers customizations if he feels they will be needed or relevant. He weighs his personal cost of putting the customizations or procedures into a form that others can use and decides whether or not it is worth it. He gave the following example as an illustration: "I wanted to include *idraw* figures in documents properly. If you go through the trouble to figure it out, you also go through an equation for the time it will save the others. In this case I miscalculated; it was not worth the time and effort because they didn't use it very much."

D3 arrived approximately one year ago. After a short period of time, he took over the role of helping people in the group with customizations. D4 describes the the change: "D3 has taken over all that [sending customization information] -- he dives in and reads the documentation.". Both D3 and D4 said that there had been no formal arrangement, but everyone in the group was aware of the change. D4 felt it was convenient to stop performing this translation role "when someone else appeared on the scene to do it...[D3] likes it and we don't need two people to do it." He describes the groups attitude as very informal: "If there's a vacuum, someone steps in to fill it." Neither D3 or D4 spend much time customizing, relative to their other work activities. Nor do they claim to enjoy it much, especially if it is forced upon them by a system change. D3 says "I like things to be predictable and out of the box -- I don't like customizing. I got so pissed off when they updated to MWM and Muse didn't run. I did Muse repairs and send a message to [the group] on 'MWM and You'." Their motives for helping others customize appear to have less to do with a desire to spread customizations, as much as a desire to protect their colleagues from spending too much time or getting frustrated. D3 now sends out mail to others about customization and helps new people set up their customization files when they arrive. The more technical members of the group tend to copy parts of D3’s files and the less technical members (E5 and E7) "take the whole file". He helped the least technical permanent member of the group (E5) get set up and D3 now feels that E5’s are the most "prototypical" for the group. Both E5 and D3 make their files "world-readable", so that newcomers and others may borrow their files. But D3 says that E5 is "too smart to lend his files to other people because he know’s he’ll have to support them." D3 is aware of the responsibility he has taken on: "Like the old Chinese proverb: once you give someone your dot files, you’re responsible to them for life."

C8, C12 and D5 are the most technical members of the group. C8 and C12 work on system issues and customize very little. D5 works on applications and has a graphic design background as well as an advanced degree in computer science. He has the most subtle set of customizations on his workstation and can articulate his reasons for each. (He has also created the most aesthetically pleasing workstation in the group.) He rarely shares these customization files; an exception is a clever variation of an M.C. Escher print that he created as a background to be interesting but not distracting. Several people in the group asked him for this background (D3,D4,A8,E7) and have created variations of it in different colors. D5 has created several very sophisticated customizations, such as placing a pixel-wide red line on the scroll bar of a dark gray window so that he can quickly distinguish remote windows from local windows, without being distracted. He does not actively offer his files to other members of the group; they only get the customizations that are obvious enough for them to see and ask for.

Because members of the group give many demonstrations, often at different workstations, they prefer to use relatively similar sets of commands. This group is very closed with respect to the rest of Project Athena. Although they actively share customization among themselves, they rarely shared customization files and ideas with others in the rest of the organization, even people with similar workstations. This behavior was also influenced by a recent change in the way protections are handled at Athena. After the system update, D3 sent out a mail message explaining what happened and how to create the protections so that other members of the group could read their files. In a list of other options, he mentioned how to make the files accessible to everyone on staff, but only he and E5 chose to do this. As D2 says "I don’t share customizations outside of the group." The reasons for this isolation are partly historical, because the group has only been recently recognized as serving a legitimate function at Project Athena, and partly due to the type of work they do. They develop prototypes of new kinds of multi-media software rather than provide an on-going service for large numbers of users. The only people who have much contact with the group are E1, who actively gets to know everyone and acts as an information resource for all of Athena, and D1, who used to be in Operations and worked on their hardware. Not surprisingly, a certain amount of resentment is expressed about this group by others at Athena, and the members are considered cliquish.

6.2.2 Making customizations publicly available

This organization has a number of extremely technical systems programmers who experiment with the system constantly. Some have an order of magnitude more customization files than the average staff member. This group tends to be young (often recent graduates of MIT) and members still find the system fascinating. They interact most with each other and the student programmers, who are only a few years younger than themselves. Members of this group like to impress each other with their knowledge of the system and their ability to make it do unexpected or "neat" things. When one of them invents something interesting, he[26] may publicize it by sending an electronic mail message, creating a zephyr message (which pops a message up on subscriber's screens) or by posting it on the electronic discussion system. Some of them also answer questions in OLC (the on-line consulting system) or on the Zephyr help instance (described in Chapter 5.) These broadcast messages may contain the customization itself. Other times, the message will tell people to look in the sender's personal files. For example, C4 recently sent the following message to the staff distribution list:

```
To: staff@MIT.EDU
Subject: Fuzzballs for the office
From: [C4]

For those who wish to still have fuzzballs on their office
signs, supply the following path in lieu of the default
"/mit/consult/lib/office/office_owl.PS":

        /afs/athena/user/u/username/data/office-fuzz.ps

I converted the fuzzball this weekend to be useful with the
"office" program in the consult locker.
```

[26] Although there have been a few female systems programmers in the past, there are none currently.

The "official" Athena owl looks like: The alternative "fuzzball" looks like:[27]

This message makes perfect sense to programmers and technically astute members of the staff. However, it is not particularly accessible to the rest of the staff. C4 is unaware of this.

He has made his software available by publicizing its location in his personal directory, which he has made "world-readabled". Any member of the staff can copy the file "office-fuzz.ps" from the data subdirectory of C4's home directory. (In this example, I replaced his actual username with "username".) This technique of providing read-only access to personal directories is quite common in this organization, especially among the systems programmers. Over one third of the staff (30/80 or 37.5%) make their files accessible to others in this way. (The individuals who make their files available are listed in Table 6-1.)

C4 is fairly typical of the systems programmers who make their files world-readable. He has created a highly-customized set of dot files and many people want to look at them for ideas. He says: "A lot of people copy files from me. Some have even taken my entire startup files." On the other hand, he says he doesn't like to be bothered by too many questions and he doesn't get involved in many exchanges of files. "I don't normally take much form others. It's just as easy for me to write it myself. I take other people's patterns, but not their code." C3 also makes his files world-readable and discourages people from talking to him directly. "Nobody comes to ask me questions, but my home directory is world-readable."

Other members of the staff appreciate this approach because it allows them to see the files without having to talk to the programmers. Many stated that they found these systems programmers difficult to talk to. For example, C11 says he never asks C4 for help directly because: "You get a

[27] I tried following these directions and created a postscript stack error. This is another example of the problem with borrowing a customization that is not fully understood.

10-minute core dump[28] instead of the answer...I want an easy way to get at the options." A2 says of C3 and C4: "You really have to pay if you want to talk to them." She contrasts this with talking to A3 and E16 "who never make you pay". The latter two enjoy helping people and don't ask for anything whereas the former two will only help if they receive a favor in return. Most people are also very sensitive to other people's time, especially other system programmers. C8 says: "I never ask people [for help with customization]. I'd be forced into it if I didn't have access to A1's files. If I didn't, I'd ask him for his dot files and not bug him." Those with with sufficient technical knowledge to a) find someone else's directory, b) find the appropriate files, and c) read, understand and copy them, can avoid bothering people and still gain access to their expertise.

The people who have their files copied the most are C1, C2, C3, C4, D6, D7 and E16. Three people, D6, D7 and E16 also spend time talking to people. The system programmers (C1-C4) are more difficult to talk to and tend to broadcast information or simply make it available via their files.

In a sense, this is a special kind of appropriation of the system by its users. Managers do not encourage or reward this activity and some are unaware that this avenue even exists. Yet almost half of the staff look for ideas and working examples from other people in this way. Many people prefer to looking in a directory than talking to a person. The key element here is to be able to give and receive useful information without incurring any obligations. The creator of the file who explicitly decides to make files accessible avoids responsibility for maintenance and does not have to explain the files to others. People who borrow files can examine a set of working examples without bothering anyone and choose whatever seems most appropriate.

People sometimes borrow files when they have an idea or a need and suspect that others have solved the problem. C5 says: "I tried to figure out the windowgram...I saw C1's and wanted a similar effect". So he copied it and spent several hours working on it. C7 says: "Occasionally I want to do something. I think 'D7 does it', so I attach his home directory and look." Another strategy is to simply go, as E10 says, "junk hunting". Several people said they attached files as a way to learn about and explore the system. For example, E14 says: "When I first arrived, I sat in on the consultant's seminars. I attached their directories and used them as a guide." E10 describes another common situation: "If I see something neat, I look at their path [access to the directory], attach the directory and try it. If it's good enough, I keep it." Other people look for files but are

[28]A "core dump" is a computer term that refers to the contents of memory at the time a program crashed. This use of the term conveys the sense that the speaker is providing completely unedited, minute highly technical details that are indecipherable to the listener.

adamant about not copying them directly. For example, E11 says: "I don’t like to just try things, I like to understand. If I copied something, I’d take too much time to figure it out...other people’s files are baroque and not logical." Instead, he when he sees something, he thinks about it, and when he feels comfortable with it, he creates his own version. He gave an example in which he said "Now it’s functionally through my head what the nature of the solution is. I was just waiting to be comfortable [with it]." E14 used to copy files, but now only uses them as examples: "I borrowed someone’s, didn’t like it and got rid of it. It was very confusing. It’s hard to use someone else’s files, because it’s hard to figure out the interdependencies. I wait for it to assimilate then do it a couple of days later. I don’t know what the ramifications are, so I don’t take anything directly."

This type of free sharing of files among consenting members of the staff has a cost: there is no built-in quality control. However, the system is somewhat self-regulating. The least technical members of the staff are unaware of this mechanism for obtaining files and thus do not copy them blindly. The rest of the staff are more likely to be like E11 and E14, who examine other people’s files for possibilities, but then create their own versions. They either come to this strategy because someone else’s files caused problems or broke or because they feel they need to understand the system and can’t simply adopt other people’s files.

Experts broadcast customization files or make them available

Some staff members, mostly programmers, also make their files available through different broadcast mechanisms. There are ’lockers’, which are like directories, that are publicly accessible and contain examples of code. The Student Information Processing Board (SIPB) also monitors files and broadcasts useful software. (This is an informal, student-run group of programmers.) Other people post messages on specialized Zephyr distribution lists, in an appropriate ’discuss’ meeting (Athena’s electronic notes file system), or via electronic mail to staff or other interested groups. Consultants identify questions that are asked repeatedly and create "stock answers" which are made available through the OLC system. The people who do this view it partly as a community service and partly as a way to get recognition for their technical prowess. They sometimes borrow files but, as C4 says: "I don’t normally take too much from others, since it’s just as easy for me to write it. I take patterns, not code."

Etiquette about borrowing customizations

This organization has established a set of social norms around the borrowing of customization files. The act of making one’s files open is viewed as an invitation to borrow the files. E14 describes the general attitude, which is, "If it’s world-readable, it’s OK to look.". As D3 says: "Anyone who wants my dot files is welcome to them." Newcomers who are technically savvy quickly learn that this is a common practice and use it as a way to learn about the system. Some people are aware of the practice, but don't engage in it. For example, C10 says: "I've never attached someone's locker to look [at files]. I understand that I can do it, but..." He then corrected himself and said he had done it once, borrowing a particular file from C3 that A4 had told him about. He has come recently from the Soviet Union and feels very uncomfortable with the idea of taking other people's files.

Most other people see it as a a socially-responsible way of sharing one's innovations without incurring the responsibility for supporting them. Thus, as mentioned earlier, C8 appreciates the fact that he doesn't have to "bug" his officemate with questions and can simply borrow the files he needs. People who borrow these files feel something of an obligation to leave their files open as well. E14 feels that "my files should be world-readable" because she's borrowed so much, even though she doesn't feel that she has much that anyone else would be interested in.

Not surprisingly, the people who do not customize much do not leave their files open. People who are very busy and not interested in exploring the system are also less likely to keep their files open. What is perhaps surprising is that some of the people who act as translators in the organization (A3, A10, D4, and E2) do not leave their files open. In D4's case, this may be because he has stopped actively sharing his files and D3, who has taken over the role, does leave his files open. Each of the remaining translators only share their files with secretaries or non-technical managers, who are not technically sophisticated enough to know how to look for and use such files.

6.2.3 An example of a translator

Several people in the organization stand out as people who actively help others customize their files. Unlike the people in the previous group, they work with the less technical (or most busy) members of the staff and try to suggest customizations that meet their needs. E1 is a good example of a someone who can translate users' needs into technical solutions. She is extremely well-liked by almost everyone in the organization. An MIT graduate who has used the system for many years, she is proficient with the system. However, her job does not require her to use much of her technical expertise. She acts as a liaison to the faculty, helping them to find what's available on the system and helping them work with other members of the Athena staff.

E1 communicates regularly with most of the staff, including the student programmers. Like A3 and E16, she is very approachable and people find it easy to ask her questions. People like to show her things as well, so she acts as an information clearing house and can help people find the person with the answer to their questions. She interacts well with the student programmers and the systems programmers. She asks the students questions as a way of showing interest in their work: "There's a lot of social pressure for the [student programmers] to do neat stuff...it becomes a religious issue. [I don't feel this pressure], but I go there and ask questions. I get their respect if I show an interest in their work and ask how to do something." She is one of the few people who talks to members of the video group described in the first case study.

She tries to avoid excessive customization and talks about two competing pressures: trying to keep standard (and thus do very little in the way of customization) and keeping on top of new things. She "satisfices", getting the system to the point where it's good enough, but she doesn't spend the time necessary to make it perfect. Even though she avoids using non-standard features where possible, she has customized her window manager to be the same as the first one she used when she started at Athena. (She received customized key bindings from D6 about four years ago and refuses to convert to the standard version. She says: "It's just the way I do it -- I'm too lazy to change. It's not that it's hard, it's just that it's not worth the effort...I go crazy when I use someone else's workstation.")

Her former officemate is now her boss and often asks her for help with the system. A good example is the Zephyr message system which was recently released with new customization procedures. In the first interview, E1 said she planned to teach herself to use the new customization format in the next few weeks. When she was satisfied that she understood it, she would help A14 get set up. She planned to take someone else's file that was "pretty close", take a break for "an hour or so" and try to make it work. At the second interview, she had done this and then continued to customize her Zephyr files because she planned to give them to the teaching assistants who were using the "on-line TA" software. Her files were to become the standard files that everyone would use.

She feels confident about her ability to learn how to customize what she wants in the system: "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." She says it's easier for her to customize than many others on the staff: "I can ask why and how." One of her strengths is that she can understand the needs of the people who ask her questions about customization and give them answers that are useful. Rather than simply telling people where to find a file, she tells them how they can use it effectively. She can be described as a *translator*, who translates the questioner's needs into something that the technology can provide.

Several other staff members perform this function as well. E16 and A3 are commonly cited as being extremely helpful, accessible, and talk at the right level. Other people go through periods in which they provide this kind of help to others. For example, D6 decided that the early versions of Athena did not make the window manager easy to use. He created a set of customizations that many staff members continue to use today (including E1). D1 and D9 are working on a system to provide a direct manipulation interface for screen layout and other customization features. In general, the projects that are viewed as successes are those performed by people who actually seek feedback from the people they help.

An important observation here is that one of the key functions of customization is not to select a static set of features, such as background pattern or font size, but to encode a commonly-used pattern of use. As in the study of Lens, reported in Chapter 3, these customizations affect the *process* of using the software. One explanation as to why the translator is so useful is that she is able to help people set up useful processes for handling their work. The selected items are not independent -- they make sense as a cohesive whole.

6.2.4 How conflicts about translation can arise

The hardware operations and administrative groups are comprised entirely of people with low technical skills. As in the video group, one or two individuals from each group volunteered to help other members of the group get started with basic customizations. The hardware operations subgroup was helped by D1, who was formerly a member of the group. An ex-MIT student, he developed sufficient technical skills to move out and become an applications programmer in the Educational Initiatives group. The administrative staff was helped by two people: E2 is a member of the training group and A10 is the manager of the group.

E2’s help inadvertently created several problems. E2 is extremely well-intentioned and feels strongly that the system is too hard for secretaries to use. He likes to share what he has learned and has strong opinions about what makes the system easy or hard to understand. Every time a new secretary arrives, he offers to help set up her workstation in a useful way. "When clerical types show up, I make a preemptive strike and give them files." He views this as a way of preventing them from getting stuck with the basic system. He selects a parts of his current set of customization files and copies them into the secretaries’ directories. "I do a tuned version so they don’t get into trouble." He used to copy his regular files for them and delete what they didn’t need. Unfortunately, this didn’t always work well: "it was more harmful than helpful". For example, he discovered that he had placed his username in one of the files. So every time one of the secretaries used a particular feature, a message was sent to him instead of the secretary. He tried to remember to search for his username when he set up subsequent files but found that he still forgot some. So he has now created a standard "cut down version" for everyone. This version uses a variable ("whoami") which generates the correct username.

Although a member of a different group (in User Services), E2 is sympathetic to the secretaries and sees their situation as being very difficult: "They never have time to learn. I make them leave their desk, forward the phone and learn what they need to know." He wants to protect them from the system and wants to ensure that they have the resources they need to learn about it. He has given his files to B1, B2, B4 and plans to give them to B3 "when she has time". He has also given his files to numerous other secretaries who have come to Athena over the past few years.

Unfortunately, the manager of the secretaries (A10) has found that E2’s efforts have created problems. "I hate it when E2 gives them something different...they get screwed up." The secretaries do not have the technical skills to make judgements about which software or software features to use. They find themselves with unexplained problems if they happen to use someone else’s workstation and most are not aware that "dot" files even exist. They have no way of changing the files if there are problems and it makes it more difficult for other people to help them. A10 has asked her staff to stop taking files from E2 (although she hasn’t spoken to him directly). They now go to her for help with their files, which are kept extremely simple and relevant to their jobs. One of the secretaries (B4) described problems that had stemmed from using E2’s files "He programmed what it does but I don’t use it much. I’ll go to Mary and ask her how to do it [to make it work the way she wants it to work.]". Interestingly, although both E2 and A10 share their files with others, neither have allowed their home directories to be world-readable. For them, sharing customization files is a personal, one-on-one activity, not an anonymous gift.

6.3 Who shares customizations?

Several people in the organization stand out as sharers of customization files. Of the managers, A3 and A10 each share files and personally help other staff members. A3 provides help to people throughout the organization and A10 provides help only to her staff. None of the secretaries actively share customizations, partly because most of them are either non-technical, new or both. Of the systems programmers, C1, C2, C3 and C4 have a big influence on the rest of the staff via their files. They never seek to help other people but will answer questions if asked. (Other people describe C1 as more skilled at interpreting questions and providing useful answers than the other three.) They sometimes broadcast notices about new customizations or provide them in public directories for others to use. Other people (particularly other systems programmers) usually obtain copies of their files by looking for them in their home directories. Of the application programmers, D1 formerly helped a group (that is not well-represented in this sample) and E13 actively helps them now. D5 and D8 each created a single item that has been noticed and copied by many people. Within the video group, D4 and later D3 actively share their customizations with the other members of the group (but not the general staff). E1 often helps indirectly, by helping people find who is likely to have the most appropriate customizations. E16 is the most active customization sharer on the entire staff. He both talks to people and makes his files available for others to copy.

Two different styles of customization sharing can be identified. The first is relatively anonymous, in which the customization is either broadcast to other members of the staff or the file is placed in an accessible location. The people who obtain customizations in this manner must be proactive and sufficiently skilled to take advantage of the files. The second style involves conversations between people as well as an exchange of files. In these situations, one person attempts to identify the needs of another before suggesting particular customization files or techniques.

6.3.1 Anonymous sharers

The first group often act as "anonymous donors"; they make their files available and have no idea who uses them. They are usually highly technically skilled people who enjoy pushing the software to its limits and react to peer pressure to "do neat stuff". Most of them are (young) systems programmers, although a few are applications programmers (e.g. D7). They have very complex sets of customizations that are of interest primarily to other highly technical people. The share the characteristic of C3 and C4 in that they have difficulty communicating directly with the rest of the staff. Because they are the most technical members of the staff, members of this group are usually the first to investigate new software packages and usually set up the default files that will be used by everyone else. This presents a problem. These people concentrate on the technology, not the use of the technology. The system defaults they create are technically sophisticated but probably not the most effective from the typical user’s perspective.

Many other members of the staff also open their personal directories and encourage anonymous sharing of files. The two managers who keep their files open (A4 and A9) have staff members who might find their files useful. Note that the managers who actively share their files (A3 and A10) do not keep their directories accessible. Most of the applications programmers (except the members of the video group) keep their files open, partly as an incentive for other people to try the software they are creating. Approximately half of the other staff members (who are not programmers) also keep their files open. Most of the system programmers and many other staff members (A1, A3, A4, A5, A6, A9, D7, E1, E3, E8, E9, E13, E14) state that they first look at other people’s files when they decide to customize. Some people copy the code directly and others only use this to get ideas or to copy the syntax. One of the advantages of opening one’s home directory, in addition to convenience, is that one can share without implying a high level of support. In contrast, people who post messages are also implicitly announcing that they are interested in feedback and will accept questions about the file.

6.3.2 Translators

The second group of customization sharers talk to people as part of the sharing process. These people are almost always less technically skilled than the systems programmers. They say they share customizations because they like to help make the software environment easier for their colleagues to use. These people translate complex files into simpler ones that provide practical benefit to the recipients. Most of them understand the basic design of the system and can talk to the technical staff (or borrow their files) if necessary. They have better general communication skills than the "anonymous" sharers and are more interested in customizations that solve practical problems than ones that demonstrate technical brilliance. They try to protect those who either don't understand or are simply not interested in learning what customizations are possible and how to accomplish them. Unfortunately, because they are less technically skilled, their customization files are more likely to contain errors. Thus, they may pass errors to their colleagues without knowing it. People in this category include: A3, A10, C1, D4 (formerly), D3, D6, E1, E13, and E16. Some of these people make their files "world-readable" and most limit their help to specific groups of people.

The first group of customization sharers appear to perform their function independently of the needs of the individuals of the organization. They do not react to reorganizations or job changes and generally broadcast customizations whenever they happen upon something interesting. In contrast, translators are very much affected by their roles in the organization. They are much more likely to remember who they gave files to and why. Most perform the role when the need arises and stop when circumstances change so as to make it unnecessary. For example, in the video group, D4 willingly gave up the role when D3 arrived. Another member of the group, D2, said "I used to spend time building an environment for beginners at [previous job], but here I've only done it with TeX files. I spent an afternoon cleaning up my files and then I introduced it to the group. So I take responsibility for only one piece of the machine."

A10 did not initially help members of her group. She felt that someone from the training staff would be likely to do the job well and referred her people there. However, when problems arose, she took over the function herself. In the operations group, D1 was the most technical member of the group and helped everyone else get set up. When he was promoted, he moved into a group in which he was of average technical ability. Although willing to help out, he does not act as a translator for this group, although he continues to help out members of his former group with questions as they arise. When he left, E13, who is in a related group, expanded his role and gives his files to people in both groups. Like E2, the files he gives often cause problems. The people who use them spend much of their customization time deleting things that are not relevant.

Table 6-2 shows the people who have acted or are currently acting as translators for different groups within the organization. The "earlier" category refers to people who performed the role of translator for a group prior to the beginning of the study. The "current" category refers to people who were performing this role during the course of the study. The reasons for the shifts from one translator to another vary from group to group. In the administration group, the manager took over this role to protect the members of her group. In the video group, the newer translator showed that he was interested in providing this kind of help and the earlier translator and he agreed informally that he should take over the role. In the hardware operations group, one person performed the role until he left, after which another person took over. It appears that only one person at a time performs this translator role for each group. However, A3 and E16 also act as translators for the organization as a whole.

Table 6-2: Translators in each group

	Translators	
Group	**Earlier**	**Current**
Adminstration	E2	A10 A3
Video group	D3	D4
Operations (HW)	D1	E13
Operations (SW)	E13	E13
User Services	E16	E16
Education Initiatives	E1	E1
Systems Development	none	none
All	E16 A3	E16 A3

Translators appear similar to the "gatekeepers" identified by Allen (1972) . 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 are unlike gatekeepers in that they are not the most technically-skilled members in the organization.

They also do not fill the role all the time; they only act as translators when there are people who appear to need the help. The differences may stem from the contrast between customization activities, which occur periodically when someone needs help getting setup or reacting to a software change, and the on-going need for accessing current technical information.

6.3.3 Creating standard customization files at Athena

Over the years, a few staff members have contributed customization files that continue to be used widely and affect the accepted norms of use. Current staff members cite one of the two creators of the X Window System and one of Athena's first release engineers as providing files that continue to influence them today, despite numerous subsequent changes in the system defaults. Two current staff members have also had major effects on how Athena customization files are used today.

D6 joined Athena about five years ago. About four years ago, he became frustrated with the lack of a reasonable set of customization files for the first customizable window manager, UWM. He says: "I have always been apathetic about user preferences -- it's one of the rat holes that absorb Athena's time and energy. They should have a good standard. But it was a different issue with .UWMRC [the customization files for UWM]. When the workstation was new, there was too high a hurdle to get up and working and productive. Athena didn't encourage people to use the workstation well. We never put serious effort into making Athena easy to use. For example, we can let people know there's a background menu, not just arcane key bindings." His own preferences had a strong influence on his choice: "I never could learn those silly bindings -- I use menus." He created a set of customization files that met his ideas of what other users would like and passed them around. Many of these customizations became part of the standard system defaults at the time and some people, including E1 continue to use them today. The 'official' version of the files has continued to evolve, but people like E1 continue to retrofit them back to be as they were before. E1 says: "I have a set of custom bindings from D6...they're ingrained...I'm too lazy to change."

E11 is a member of the documentation group who has been with Athena for a year and a half. He likes things neat and well-ordered and did not like the relatively random way in which Athena's customization files were ordered. As Athena had evolved, various defaults for customization files had been tried and discarded and the result was that everyone had a different strategy for organizing these files. He created, with help from another staff member, a set of dot files that each had a specific purpose. Thus the '.aliases' file contains aliases and the '.startup.X' file contains the commands that are run when the user logs in to the X Window System. He says: "The dot file change made [running an X session] reasonable...Now it supports the new user cleanly and the logic is clear. If [his partner] and I hadn't done it, it would have been haphazard."

The Fall release of the software contained his set of dot files and he wrote a guide to explain how they work. Some people liked the change and others found it confusing. Some people used it as an opportunity to reflect upon their use of customization files and upgrade them. Others, including E11, simply allowed their old customizations to go away: "I used to have more [dot files] in the old days...but they broke up in the Fall release." Most of the older staff members did not completely reorganize their files into the components suggested. However, the new staff members do not question this arrangement and do not appear to have any particular trouble organizing their files in this way.

6.3.4 Who doesn’t share?

Two groups of people never share their customizations and in fact, do not customize much at all. The first group consists of the secretarial staff who are generally new to the organization, have no programming skills and a busy job that does not leave much time for exploration of the system. Surprisingly, the other group is comprised of the most senior technical staff; the system programmers with 15 or more years of experience (C9, C10, C11, C12 and A1). These people customize very little. They are not impressed by the young systems programmers’ proficiency with the system and feel no need to impress others with their own programming skills. They generally work at the "edge" of the system, which means that it changes rapidly. Having been through many changes, they only perform customizations that allow them to preserve some stability across hardware platforms. They assume the system will change and do not want to invest in customizations that they will have to change later. Given this attitude to customization, it is not surprising that they do not share the few customizations that they do create, nor do they make their files accessible to others in the organization.

6.4 Why share customizations?

People cite a number of reasons for borrowing customizations. The most common reasons are to reduce learning time and to maintain consistency with others in the organization. D2’s reason was more focused. He works closely with D4 and wants to be able to make it easy for D4 to use his workstation. So, he borrows most of D4’s customizations (particularly those that affect the process of using the system) to make it easy for D4 to use his environment. "I copy D4’s stuff because we work together. Then, when he comes over, we have the same environment." It is an explicit, if unspoken, agreement between the two to facilitate their cooperative work. Another important reason cited for borrowing customizations is to get good ideas and good strategies from people they trust. For example, E11 is reluctant to borrow anyone’s files, but makes an exception for an ex-staff member who he trusted. Unlike the rest of the staff, the most technical people tend to borrow files to see an example of the syntax rather than as a source of ideas about reasonable combinations of preferences.

The people who actively share files with others appear to do it primarily for social reasons. It is important to note that neither form of sharing is officially recognized or sanctioned by any of the managers. Although most managers know that some sharing occurs, none are unaware of the extent. People who share do so on their own time and do not receive external rewards or better reviews. In fact, in one case, A2, who is A3’s manager, has downgraded her slightly because she feels that the sharing detracts from her ability to perform her other work. A2 is reluctant to actively discourage A3, partly because A3 wouldn’t like it and partly because it would make her unpopular with others in the organization. (A3 is well-liked and has a reputation as being very approachable; people appreciate her efforts. This appreciation is probably more important to A3 than A2’s mild disapproval.) Notice that part of the problem is that A3 helps people throughout the organization, not just A2’s group. Thus, A2 is indirectly providing a service, through A3, to the whole organization which is not recognized or rewarded. A8, on the other hand, actively encourages sharing within the group. In fact, he encourages D5 to share more than he does. However, A8 does not encourage sharing outside of the boundaries of the group which reinforces the isolation of his group from the rest of the organization.

All of the translators talk about wanting to help others deal with the complexity of the system as their primary reason for sharing files. They all enjoy talking to and working with others. Two people, E2 and E13, have reputations for liking to share but causing problems when they do. Thus, the only people who borrow from them are people who are new to the organization and unaware of their reputations. The people who "hack" the system also seek social rewards, but more for recognition of technical skill or clever ideas. The highly technical people who do not share much also say that they do not particularly care about their reputations for such things. A few people have very practical reasons for sharing. For example, A10 became concerned when her staff found it difficult to use the files that E2 gave them and she took over the role herself.

Different groups react differently to customization and sharing of customizations. The young systems programmers view customization as a creative outlet and as a way to demonstrate ’cleverness’. Perhaps in response to their eagerness, most other staff members downplay their customization activities. C11: "Maybe I’m just too old, but I’m totally unimpressed by the kids who are [student programmers]...Their job is to play with the system and generate useful stuff as by-products of that play. [My boss] doesn’t pay me for that." Too much customization is viewed the same way as playing too many computer games -- the person should be getting work done. A8 says that customization is "the last thing on the queue. I feel guilty doing it. I feel that I should be doing something useful like testing an application."

Members of the hardware operations group were quite reluctant to be interviewed about their customization activities. (Several participated in the first part of the study but did not agree to a full interview.) Most of these people had received files from D1 earlier and have not customized since. In contrast, the members of the video group customize, but tend to view it as a necessary evil. D3, who has taken on the role of providing customizations to others in the group says: "I like things to be predictable and out of the box -- I don’t like customizing. The pain I experience with something causes me to sit down for two hours with the documentation [and try to fix it]." However they also view it as a creative outlet. As D3 also says: "This office is quite quirky -- I like my workstation to be like this office. Some days I have different backgrounds..."

6.5 Which methods of sharing are most effective?

Different people find different methods of obtaining customizations more effective. Not surprisingly, the least technical members of the staff either ask for help from a person or they don’t customize. They never look at other people’s files and generally prefer to get help "in person" than through other means. (The only exception is OLC, the on-line consulting system, which allows users to ask questions or browse through sets of stock answers.) These people feel limited by the system, which they view as very powerful and somewhat overwhelming. They limit their questions to items that are critical, so as not to bother their colleagues. Several people in the organization look out for these people. Some are very good at translating their needs into appropriate customizations and these people provide the most useful customization advice. Others do not examine the consequences of their advice and may end up causing more harm than good.

Members of cohesive groups like to share customizations with each other, especially if their work causes them to spend time looking at each other’s screens while their colleagues are working. Because they work closely together, the most effective customization sharing comes from other members of the group, either in response to a system change that requires action or as part of on-going exchanges of information.

Technically proficient people find that making directories accessible is very useful, because they can make files available without an implied level of support and they can look for files without bothering the author. They like to borrow files or ideas for customizations and try them when it is convenient. Sometimes they make improvements to the files, which they make available to others in the same way. If someone creates something that is particularly interesting, it usually becomes known through word of mouth. The following were all identified as methods of obtaining customizations:

1. Walk by and see someone else's screen and ask. (e.g. screen background).
2. Watch someone performing a task and ask about a method they use.
3. Someone offers to help you get set up.
4. You ask someone to help you get set up.
5. You have a problem or question and ask someone.
6. You send a question to OLC or the Zephyr Help Instance.
7. Someone has a new idea and tells you about it.
8. New ideas are posted in a common area and you look.
9. You read electronic mail about an idea.
10. Someone tells you to look in the common area.
11. You have a symbolic link to someone else's file and it's automatically updated and you get the change.
12. You experiment and create it yourself.
13. You get the standard file and use it.

6.6 When does sharing occur?

Almost everyone engages in sharing software when they arrive as newcomers to the organization. Even experienced programmers usually ask for help "getting set up". For example, D6 had been a member of the organization since the beginning and had created a number of customization files that other people had used. He then left for a year and returned just prior to the reorganization. Upon his return, he asked E1 for her files, to help him get set up again. Since he had given her files in the first place, he was able to come back up to speed very quickly. People allocate the most amount of time to customization when the start a new job in the organization. They have less work to do and many describe customization as a way to learn about the system. This presents a problem, because they make decisions about their future use of the software at the point at which they know the least, both about the technology itself and their expected use of it. By borrowing customization files from other people who use the system regularly, newcomers are able to reduce the learning time and take advantage of what other people have learned. They also unconsciously adopt the standard patterns of behavior of others in the organization.

Sometimes, sharing is serendipitous. It can occur when someone walks by someone else’s office and sees something "interesting" on the workstation or it may happen while working on someone else’s workstation for another reason. In such cases, people often ask how "it" is done and borrow the customization.

New software releases also occasion sharing, simply because they force people to reconsider their current set of customizations. If one person discovers how to make a useful kind of adaptation, he or she will often share it with other members of the staff, or more likely, other members of the immediate group. During this study, Athena released a new window manager as the default and changed the customization techniques for the Zephyr message system. As described in Chapter 5, 63% of the staff retrofitted the new window manager to be like the previous window manager. Although some people were able to do this on their own, many others were forced to ask for help. For example, when the new window manager first appeared, D4 said: "I hate having MWM forced on me -- I hunt it down and kill it with extreme prejudice". He sent a message to the members of the group with instructions on how to avoid the conversion and continue using the old window manager. Two months later, everyone converted to MWM because D3 had figured out how to preserve their previous customizations in the new environment and sent out a message explaining how.

In this study, office changes do not appear to have much effect on customization in general, because the user’s software environment remains unchanged even when moved to a different machine. However, officemates are more likely to share customizations, so changing officemates increases the likelihood that sharing will occur between those two people. Job changes and the reorganization also increased the level of sharing because people sought help from their new peers. Of 77 exchanges in the pre-reorganization groups, 44/77 (57%) of the sharing occurred within a group. Of the remaining 33 exchanges, 21 were carried out by five translators: A3, A10, E2, E13 and E16. Managers and secretaries tend to ask for help from people on the staff. Systems programmers generally only get files from other systems programmers, although their files are used by people in all job categories. Applications programmers get files primarily with other applications programmers and tend to give files to people in other job categories. The people in the "other staff" category comprise a variety of jobs and they exchange files with people from a variety of categories.

6.7 Which customizations are shared?

Some customizations are more popular than others. For example, many people use a standard startup procedure for laying out their screens. Very few people copied their screen layouts directly from someone else although they borrowed parts of the setup (such as backgrounds or items such as the on-line dictionary). The only exceptions were a few non-technical staff members who did not know how to record the screen layout. When asked about their screen layouts in the interviews, staff members spoke about the design of screen layouts as a highly individual activity. Most people described the following process for designing screen layouts: begin using the system and arrange the windows and programs by hand. Try out different patterns and determine which appears most useful. When the layout seems reasonable, encode it in the startup file and set it up automatically upon login. (This may be a one-step or a multi-step process, depending upon whether the person wants to avoid slow startups.)

Based on these interviews, it would be easy to conclude that there is no sharing involved. However, when I examined the actual screens, I found 40/51 people use the same basic layout: two "tiled" columns of windows, side by side on the screen. Of those who do not use this layout, seven are members of the video group. There are variations in the layout. A few people layer several windows on top of each other within each column and others use half-height as well as full height windows. 36/51 create a space in the corner for icons, clocks, loads, etc. Of these, 29/36 placed icons at the bottom icons at the bottom of the screen and only two people (C6 and A5) placed icons in the upper left hand corner. A few people left a blank column between the two columns of windows for icons. (C2 noted that this had been an important screen layout when the first version of the window manager was released, because it was necessary to provide an uninterrupted column of background pixels on the screen to prevent "black holes" from destroying windows.) Nobody on the staff uses the default system layout, which consists of a single white window in the upper left of the screen, a console window in the upper right, and a logout button in the lower right.

Even the non-conformists didn't create substantially different layouts. E3 and A9 use two columns of windows, but they overlap in the middle. E2 and B5 use one window at a time and place it in the center of the screen. A1 uses two screen layouts. His regular layout follows the same two-column pattern that the others on the floor use. The other one places windows diagonally across the screen. He borrowed this from a person who briefly worked in the video group and the only other person who uses this layout is D2, a senior member. For all of their sharing of other customizations, the video group does not share screen layouts. Only one of the eight members of the group uses the two column screen format and the rest have layouts that differ from each other. No two are alike.

Why are the screen layouts so similar? One explanation might be that they have all somehow come across the optimal layout for managing "screen real estate". However, this seems unlikely. Most people did not consider layouts that differed significantly from the standard layout. Most people spoke about deciding which corner the icons should go in, rather than what the most effective access to the windows might be. The video group spends by far the most time thinking about and experimenting with effective screen layouts. They do not find the "standard" layout to be optimal. A better explanation is that people were subtlely influenced by simply seeing each other's screen layouts.

More explicit sharing of customizations occurs when someone solves a common problem and the solution is made available to people. Some customizations can be traced back to the earliest versions of the system, in 1984. For example, specialized commands for moving to a directory and returning to the previous directory were created by an early group member and made available to all users in 1984 and 1985, before they were removed. Note that these facilitate the process of moving through the system. People who have these items now would have had to obtain them either from the original files (if they were here then) or through copies of someone elses files. E1 and D6 both have these files. A few people remember the person who originally created the commands; most people simply copied them from other people over the years.

Background patterns and screen savers are also popular items to share. A library of patterns is available for people to browse through and choose among. Some people create their own patterns and share them with others. (For example, D5 shared his Escher background with other members of his group and several people outside the group borrowed it as well.) Very popular items are added to the general library of background images. These are among the easiest to share, because they are easy to see and easy to copy.

Users in this environment often borrow customization files from other people. As mentioned earlier, one of the common components of a customization file is the encoding of a common pattern of behavior or a common collection of activities. Borrowing these usage patterns may affect the recipient for many years. For example, the members of the hardware part of the operations group continue to use the standard setup file that D1 created for them several years ago. The members of the software operations group, who borrowed E13's files, found that they were immediately subscribed to a large number of Zephyr lists. They were influenced both by the traffic and information content of these lists and all of those interviewed (E9, E10 and E14) removed themselves from most of the lists. However, the members of this group have a higher number of Zephyr subscriptions than any of the other non-technical groups that did not borrow customization files containing subscriptions. (These people do not have a job-related need for extra Zephyr subscriptions.)

The members of the administrative staff use the limited subset of commands and applications provided by E2 and A10. The members of the video group express their individuality in some ways, such as highly divergent screen layouts and use of color. On the other hand, they are all very much influenced by each other and have very similar work environments. As mentioned in Chapter 5, D2 and D4 try to keep exactly the same set of working commands. Because the members of these groups all perform related functions, it is difficult to disentangle the effect of providing standard customization files from the needs of the job. However, given that most people tend to settle on the customization files they start with, with small variations over time, and that there are a wide variety of options in how to organize work environments, it appears that the exchange of customization files is serving to reinforce group norms of behavior.

Chapter 7

Revisiting the Structurational Model at Project Athena

The data presented in Chapter 5 and Chapter 6 can be related back to the structurational model described in Chapter 2. In this chapter, I analyze how the behavior of individuals in the organization is affected by institutional properties, the changing technology, other external events, and factors specific to the individual. I also examine how the behavior of these individuals affects their future customization behavior, the technology, and the organizational norms. I include one case in which customizations by the users of the technology resulted in a new model of use, which affected the subsequent development of the technology, which further affected both individuals and organizational norms. Figure 7-1 illustrates the relationship between the customization behavior at Project Athena and structurational model.

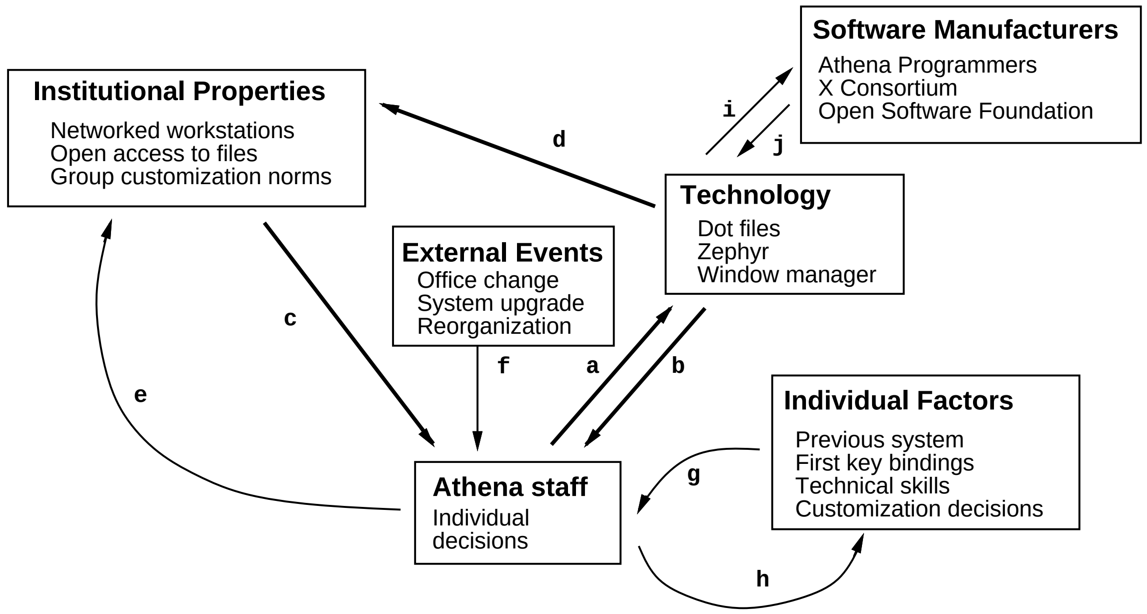


Figure 7-1: Structurational Model: Athena example

7.1 Arrow *a*: Users appropriate Unix through customization

Chapter 5 identified several examples in which people in the organization appropriated the software through various customizations. Some of these appropriations affect a single individual's use of the technology. Others have had an effect on the entire organization. In several cases, the appropriations by individuals in this organization appear similar to those found in the Lens study. The following are examples of the ways in which users and groups of users appropriated the Athena environment through customization:

Video image background:
D1 created a new method of creating background screens by taking a video input from a camera and capturing the image. (Chapter 5, Page 98.)

History files:
A1 and C5 created a new use of the history file, making it act as a menu of the 1000 most recently-typed commands. These users prefer to spend time scrolling through a list to see a command they recognize rather than recall the command syntax and retype it. They find aliases too "permanent". (Chapter 5, Page 96 and Page 113.)

Zephyr alarm clock:
C1 uses Zephyr as an alarm clock with a "snooze alarm". He specifies a time to send himself 500 messages, each with a beep. (Chapter 5, Page 114.)

Screen layouts:
Many users created context-dependent screen layouts which include a set of programs that are approriate for a particular work situation. (Chapter 5, Page 5.5.5.)

Encode patterns:
Most users who customize discover that they repeat certain patterns of behavior and then create an alias or use some other mechanism to encode the pattern. (Chapter 5, Page 112.)

User-created help:
Students created a separate subscription list (called the "help instance") for peer-to-peer consulting. (Chapter 5, Page 116.)

Zephyr locations:
Users specified separate screen locations for personally-addressed messages, to distinguish them from messages sent to a group of people. (Chapter 5, Page 5.5.5.)

Table 7-1 shows two examples of how users affect the technology, broken down by job category. Column two shows the number of participants in each job category. Column three shows the percentage of participants who use aliases as a way to customize their software environment. Approximately two thirds of the participants use them. None of the secretaries do, because they are unaware that they exist. Of the remaining staff members, the applications programmers are most likely to create aliases. Column four shows the percentage of people who use a significant number (over 80%) of the aliases they create. Although the applications programmers are the most likely to create aliases, they are also the most likely to forget about them. Only a third of the non-technical staff use most of the aliases they create and only 20% of mangers and 25% of the systems

Table 7-1: Users modify the technology

Job category	N	Use any Aliases	Use 80% + Aliases	N	Retrofit New WM	Keep Old WM	Stay Same
Managers	10	60%	20%	8	63%	0%	63%
Secretaries	5	0%	n/a	1	0%	0%	0%
Systems Programmers	12	67%	25%	11	55%	27%	82%
Applications Programmers	8	88%	13%	7	71%	14%	85%
Other staff	16	68%	33%	13	62%	23%	85%
Totals:	51	63%	22%	40	60%	18%	78%

programmers do. People use aliases to modify the software in simple ways, but then often forget about those modifications.

Columns five through eight show how users reacted to a major change in the window manager. Column five shows the number of participants in each job category. Since 11 people were new to the organization, they never learned an earlier window manager, so they are not included in this table. All of the people who had customized the previous window manger in some way (78%) managed to find a way to maintain their existing patterns of working. 60% used the new window manger, but retrofitted it to keep their old customizations. 18% simply refused to use the new window manager and modified the startup procedure so they could use the old one. 63% of the mangers maintained stability by retrofitting the new window manager to be like the old one. The rest simply learned the new window manager. In contrast, 82% of the systems programmers and 85% of the applications programmers and the other staff found a way to keep the same interaction with the window manager. Of these, the systems programmers were most likely to simply refuse to use the new window manager and the applications programmers were most likely to retrofit the new window manager.

7.2 Arrow *b*: Users are affected by customizations

The Athena system provides a complex, highly customizable multi-tasking environment. Although there are many possible patterns of use, any individual will only use a small subset. The choice of that subset is very much influenced by the user’s previous experiences, their technical skills and how others in the organization use the software. Characteristics of the software environment affect how users think about and interact with the system. User’s previous choices about customizations, including the ones they borrowed from others, continue to affect them. The following examples describe how the technology and the use of it at Athena affects user's behavior.

Shared layouts:
Most of the members of the staff use the same basic screen layout (except for the members of the video group), although they are not aware of it. It appears that they been influenced by other people's screens, which affects what they think of as a "normal" screen. (Chapter 6, Page 152.)

Copied patterns:
When people, especially newcomers, adopt other people's customization files, they also adopt corresponding patterns of behavior. This serves to create and maintain group norms of behavior. (Chapter 6, Page 153.)

Past experience:
Users who join the organization with extensive experience with one or more previous systems are likely to customize the Unix environment to be more similar to the old environment. (Chapter 5, Page 118.)

Table 7-2: The technology influences the users

Job category	N	Learned new window manager	N	Influenced by previous system
Managers	8	60%	10	40%
Secretaries	1	0%	5	0%
Systems Programmers	11	67%	12	67%
Applications Programmers	7	88%	8	50%
Other staff	13	68%	16	16%
Total	40	63%	51	37% (7 operating systems & 5 applications)

Table 7-2 shows how users are affected by the technology, broken down by job categories. Columns two and three show the percentages of each job category who were forced to learn the new window manger when the system changed. None of these people had customized the previous window manager and all used the default version of the new window manager. Columns three and four show the percentages of users who specifically mentioned that their customizations were influenced by a previously-used system. They mentioned seven different operating systems and five different software applications. Two thirds of the systems programmers and half of the applications programmers performed this type of customization. Not surprisingly, the people with less experience with other systems, secretaries and other staff members, were least likely to perform this type of customization.

7.3 Arrow *c*: Institutional properties affect customization decisions

Chapter 6 describes at length some of the patterns of sharing of customization files within this organization. Some of the social rules about this sharing are very informal. For example, members of the staff understand that people who leave their customization files 'open' are inviting others to borrow them, even though there is no formal policy on this and many of the managers are unaware that this activity occurs. Managers are more aware of other activities, such as the role of translators who explicitly share software with others in the organization. Some managers encourage this, as A8 does with the video group, and others discourage it, as A2 does with A3. The following are examples of how some of the institutional properties influence users and the customization decisions they make.

Social pressure:

Different groups encourage or discourage customization activities, e.g. the operations group discourages customization and the systems development group encourages it. (Chapter 6, Page 148.)

Etiquette:

Most users are sensitive to their colleagues and several unspoken rules of etiquette about borrowing customizations are apparent. Most people do not want to bother their colleagues more than is necessary. Keeping files open is one low-cost way to provide access to customization files. Translators also try to make it easier for their colleagues to customize. (Chapter 6, Page 138.)

Table 7-3: Group norms affect users

Job category	**N**	**Borrow Files**	**Same Screen**	**N**	**Share Office-mate's Files**
Managers	10	2.6	80%*	5	40%
Secretaries	5	3.2	80%	2	100%
Systems Programmers	12	1.8	92%*	12	42%
Applications Programmers	8	1.9	50%*	7	57%
Other staff	16	2.3	81%*	11	91%
Totals:	51	2.3	78%	37	62%

* indicates that all "novel" screens in this category are from the Video group

Table 7-3 shows how others in the organization affect individual users. Column two shows the number of participants in each job category. Column three shows the mean number of people the participants borrow files from (either by copying files or working with someone in person).

Secretaries borrow files the most, followed by managers and other staff. The programmers are least likely to borrow files from other people. Column four shows the percentages of each group who use the same basic screen layout. Overall, 78% use the same basic screen setup. Seven of the eleven participants who do not use the same screen layout are members of the video group (indiciated by an asterisk), including all of the remaining applications programmers. Column five shows the number of people in each job category who have an officemate and column six shows the percentages who share. The non-technical staff members and the secretaries are very likely to share with their office mates (even if the other person is not very technical). The reluctance to share by systems programmers is supported by the relatively small percentage (42%) who share with their officemates. Managers are also less likely to share with their officemates (often because they have someone in their group who provides customization files for them).

7.4 Arrow *d*: Customizations affect institutional properties

Chapter 6 describes the ways in which customizations are shared within the organization. Because these files continue to be used, sometimes for years, one person can influence a large number of people without being aware of it. The following examples describe events in which customizations by an individual or small group resulted in changes in the norms defined by the institution.

Consulting process:

The creation of the ’Help’ list by a group of students changed the way the consulting organization works. (Chapter 5, Page 117.)

Donated standards:

Over the years, various individuals have donated their customization files for use as system-wide defaults at Project Athena. In each case, these files have become an important part of the customization files used by everyone and even when the system-wide defaults change, many people continue to use these files. (Chapter 6, Page 146.)

7.5 Arrow *e*: Customizations affect group norms of behavior

As described in Arrow ***f***, customization activities by individuals can be incorporated into the formal mechanisms for distributing software by the organization, and thus have an effect on most or all people in the organization. In Arrow ***e***, the effects have less to do with the officially-sanctioned configuration of the software and more to do with the effects of informal sharing on the the behavior of groups in the organization. The following examples describe this phenomenon.

Broadcasting files:

Very technical "lead users" of the software are the first to create customization files and make them available to the rest of the staff, usually by making their files accessible. (Chapter 5, Page 138.)

Translators:

Translators work with other members of the organization to create individualized customizations that are appropriate for the individual user's needs. (Chapter 5, Page 144.)

Table 7-4: Individual users affect group norms

Job category	N	Lend Files	Open Files	Percentage of Translators
Managers	10	1.4	20%	20%
Secretaries	5	0.6	0%	0%
Systems Programmers	12	2.3	67%	8%
Applications Programmers	8	3.8	63%	38%
Other staff	16	2.1	50%	25%
Totals:	51	2.0	45%	20%

Table 7-4 shows how actions by individual users can affect other members of the organization and thus group norms of behavior. Column two shows the number of participants in each job category. Column three shows the mean number of people the participants lend files to. (This column can be compared to column three in Table 7-3, which shows the mean number of files borrowed for each job category.) Secretaries rarely lend files, except to each other. Managers also do not lend files often (except for the two managers who also act as translators). Interestingly, the people most likely to share are the applications programmers. The systems programmers comprise two groups: the younger ones share and the more senior ones do not. Column four shows the percentages of people who keep their files open. Note that 45% of the staff keep their files open. This is partly due to peer pressure to do so. The programmers view this as an important way to contribute their files to the rest of the organization without having to support the files. Many of the other members of the staff do it because they take advantage of other people's open files and have an obligation to make theirs available as well. The secretaries are unaware that this is even a possibility and most of the managers do not keep their files open. (Note: the managers who keep their files open are not the translators.)

Column five shows the percentages of translators in each job category. Two of the managers act in this role and none of the secretaries do. The group most likely to act as translators are the applications programmers (including the less technical ones) and the non-technical staff. The systems programmers, with one exception, do not act as translators.

7.6 Arrow *f*: External events affect user's choices

External events often play an important role in determining when people decide to customize their software. External events are sometimes related to changes in the organizational structure, such as a reorganization or an office change. They may also be the result of a change in the software, either a failure or a software upgrade. Although I didn't measure it, it is also possible for changes to occur as a result of discussions with the researcher. (However, several people, while examining their alias files to see what percentage they actually use, decided to delete obsolete aliases during the interview.)

Technology changes:
Changes in the technology, either through the release of new software or upgrades in the existing software, may cause users to make additional customizations. Sometimes the changes are necessary, either to fix something that is broken or to keep a familiar behavior pattern. Other times, the changes result from the user reflecting upon his or her use of the software and thinking about how to improve it. (Chapter 5, Page 102.)

External changes:
Reorganization, office changes and job changes also influence users and may cause the user to create new customizations. (Chapter 5, Page 103.)

7.7 Arrows *g* & *h*: Individual factors and user's choices

Chapter 5 describes the different reasons users cite for customizing and the different circumstances under which they do it. Examples of factors that are under the user's control and influence the user's choices are:

Past choices:
Once a user makes a customization decision, the resulting change will continue to affect his or her interaction with the software. Since many of these customizations are borrowed from others, users may be unaware of how the customizations will affect their behavior. (Chapter 5, Page 118.)

Technical skill:
The user's level of technical skill plays an important role in how easy or hard it is to make certain kinds of customizations and thus influences the users perception of the relative cost of making the customization. (Chapter 5, Page 107.)

Experimentation:
Some users enjoy experimenting with the system. While a majority do this when they first learn a new software application, most people stop this after they become familiar with the software and only a few continue to explore. (Chapter 5, Page 108.)

Maintaining stability:
The most common form of customization is to maintain familiar patterns of behavior in the face of change. Some users are adamant about not learning new patterns, while others view it as a necessary part of working with new software. (Chapter 5, Page 117.)

7.8 Arrows *i* & *j*: Influences on future development

Although less clear than in the Lens study, user's appropriations of the technology through customization influenced future software development on a number of occasions. The creation of the Zephyr help list, described in Section 7.1, is also an example of how changes instigated by users became part of a future release of the code. This in turn allowed a significantly greater percentage of Athena users to use it. Sets of customizations by individuals E11 and D6 were codified and added to the system, based on their individual (and untested) preferences. Later designs of customization features incorporated items that users either discovered or requested.

7.9 Summary

This chapter describes how the activities of the members of the Athena staff fit into the structurational model. Users were both influenced by the software as it and their jobs changed over time, and also appropriated the software to different extents. Table 7-5 summarizes the results of this analysis.

Table 7-5: Summary of Arrows in Structurational Model

Arrow	*Definition and Examples*
a	**Users appropriate Unix through customization**
	Zephyr Help list for peer-to-peer consulting History files as a menu of command alternatives Aliases to encode common work processes Context-dependent screen layouts Video image background Zephyr alarm clock Opening customization files for others to share
b	**Users are affected by customizations**
	Unconscious sharing of screen layouts Copying and use of standard patterns of behavior Influence of past experience with other technology
c	**Institutional properties influence customization decisions**
	Social pressure about customization Etiquette about borrowing customizations
d	**Customizations affect institutional properties**
	Zephyr changes affect consulting process Donation of customization files to Athena
e	**Customizations affect group norms of behavior**
	Experts make available or broadcast customization files Translators develop individualized customizations for others Small groups share customizations
f	**External events affect user's choices**
	Changes in the technology: system upgrade Reorganization Office change
g & h	**Individual factors and user's choices**
	Influenced of past choices and experiences Level of technical skill Willingness and desire to experiment Desire to maintain stability
i & j	**Influences on future development**
	User innovations affect future design features

Chapter 8

Discussion

This dissertation postulates that the general notion of co-adaptive phenomena applies to the use of information technology. These data demonstrate that people are both influenced by the software they customize and at the same time appropriate the software for their own purposes. These appropriations may be shared and have the potential to influence group norms of behavior, other more global institutional properties and future implementations of the technology. Customizable software provides a particularly good testbed for studying co-adaptation, because users create artifacts when they customize that continue to affect their behavior over time. These artifacts may be studied independently and may be traced as they are shared by others in the organization. The presence of sharable artifacts makes it easier to study customization than related topics such as learning and innovation. Current research models do not provide useful ways of exploring co-adaptive phenomena, thus requiring new research models and methods. The structurational model, borrowed from organizational theory, provides a useful structure for studying co-adaptation and this work extends the structurational model by providing evidence of how the behavior of individuals can influence the organizational norms.

I have two primary interests in this research. The first is to further our theoretical understanding of how people interact with technology over time. This area of research has a long-standing theoretical tradition (Levitt and Whistler, 1958, Markus and Robey, 1988), but there is still a great deal that we do not know. Much of the work has been at the macro level, although according to Barley (1990), "microsocial research on technological change has recently enjoyed something of a resurgence." My research and primary theoretical contributions are on micro levels of analysis, but they are informed by the macro analyses and provide additional insights by demonstrating how individuals can affect the organization’s interaction with technology. I provide data that supports the existence of co-adaptive phenomena and claim that such phenomena require different research strategies than have been traditionally used to study the interaction of people and technology. In particular, the research strategies that treat the technology as a static, independent variable and measure change in the organization or those that treat the organization as the independent variable and measure change in the technology are each inadequate. In each case, "controlling" one of the two variables prevents the study of the phenomenon of interest, co-adaptation. I am interested in examining the on-going interactions between individuals and the technology they use and appropriate, and must therefore use different methods and models to study how each affects the other.

My second interest is a practical one. I am interested in how to improve the effectiveness of people's use of complex computer technology within organizations. This is particularly relevant given the recent findings that computer technology may have actually reduced productivity for some white collar workers (Dertouzos et al, 1989). Software manufacturers have attempted to address this in a number of ways. One has been to create software applications, such as Hypercard, Notecards, emacs and the X Window system, that are explicitly designed to be customized by end users. What is the impact of specifically allowing and encouraging users to customize software? Do people take full advantage of customization features? What kinds of effects does customization have on productivity for individual users? What are the effects on the organization? What is the importance of different features, such as those those that permit sharing among members of groups? There has been a recent surge of interest in this area, although the most of the articles are from this past year. (Trigg et al., 1987, Fischer and Girgensohn, 1990, MacLean et al., 1990, Henderson, 1990, Nardi and Miller, 1990).

This dissertation addresses both the theoretical and practical concerns. These data provide evidence that use of customizable software is a co-adaptive phenomenon and they provide an opportunity to explore some of the implications at both the individual and group level. The participants at both research sites may be considered lead users of technologies that promise to become wide-spread in the next five to ten years. The behavior of the participants at each of the research sites provide a preliminary view of some of the patterns of customization behavior that may become prevalent in other organizations. I analyze the factors that influence people's customization decisions and the effects of their behavior on others in the organization. I then provide concrete suggestions for software manufacturers, managers and end users to help them counteract the tendency to stagnate and increase both productivity and the sharing of innovations within organizations.

Chapter 1 identified a set of questions and issues about co-adaptive phenomena and customization, listed below. This chapter addresses these questions, suggests both theoretical and practical implications, discusses the limitations of the research and suggests directions for future research.

1. Theoretical questions:
    a. Does use of customizable software constitute a co-adaptive phenomenon?
    b. Does the structurational model provide useful insights?
    c. How can this analysis enrich the structurational model?
    d. How are people influenced by the software environment?
    e. How do people appropriate customizable software?

f. What are the implications for organizational theory?

2. Practical questions:
    a. Which users customize and under what conditions?
    b. How do individual customizations affect group norms?
3. Implications:
    a. What are the implications for the software development process?
    b. What are the implications for software design?
    c. What are the implications for managers?
    d. What are the implications for end users?

8.1 Theoretical questions

8.1.1 Does use of customizable software constitute a co-adaptive phenomenon?

Users in both the Lens and the Athena study were influenced by the design of successive versions of the software and their customizations of it. Users at both sites also appropriated the technology, adapting it in unexpected ways to meet individual and group needs. Both processes occurred over time and affected each other. Thus, a decision by the user to customize at one point affected later use, opinions about the software, and future customization decisions. By treating both the technology and the actions by individuals in the organization as variables, both aspects of the co-adaptive use of this technology were evident.

8.1.2 Does the structurational model provide useful insights?

The structurational model identifies four kinds of interactions that are likely to occur when technology is introduced into an organization. These are: the effects of human action on the technology, the influence of the technology on human actors, the influence of the technology on properties of the institution and the effect of these institutional properties on the activities of humans in the organization. The structurational model provides a way in which to view the complex activities of people within and organization and encourages the researcher to look for evidence of all of the elements. Chapter 3 and Chapter 7 describe how each of the relationships identified in the structurational model occurred at each research site and influenced how users learn about and customize their software environments.

8.1.3 How can this analysis enrich the structurational model?

The structurational model provide a macro-level view of the interaction between users and a new technology, although it includes the influence of individual human actors. This research provides a micro-level of analysis and concentrates on the problem from the perspective of the individual user. I added arrows to include the influence of other factors on the behavior of individuals. I added factors that influence individual's decisions, including external factors that cause users to reflect upon their behavior and change it, and individual factors, such as past customization decisions, which affect future choices. Another arrow indicates that individuals are not just influenced by properties of the institution; in some cases the actions of individual users may also influence the organization. The additional arrows highlight the existence of co-adaptive behavior by individuals in the organization. The theoretical linking of co-adaptive phenomena and the structurational model provides evidence for a mechanism by which individual interactions with technology affects both the technology and organizations that use it.

8.1.4 How are people influenced by the software environment?

Co-adaptive phenomena involve both the influence of the technology on users and the ways in which users adapt or appropriate the technology. These data indicate that users are greatly influenced by the general software environment and their own own or borrowed customizations. In both the Lens study and the Athena study, users are influenced by the preexisting software environments. One of the reasons that the first group of Lens users rejected the first prototype of Lens was that some of the features were slightly different from their regular mail system (Chapter 3, Page 53). Similarly, in the Athena environment, users were influenced by use of previous systems and applications and customized the Athena environment to make it more like the previous environment (Chapter 5, Page 5.5.6). In both studies, changes in the software and the organization affected many user's behavior. For example, one user tried Lens and discovered which distribution lists were important and which were not. He then removed himself from the distribution lists, which reduced his need for Lens, and he stopped using that as well (Chapter 3, Page 51).

8.1.5 How do people appropriate customizable software?

To be co-adaptive, the users must also appropriate the software for their own purposes. Section 3.3.2 in Chapter 3, Section 5.5.5 in Chapter 5 and Section 7.1 in Chapter 7 each describe how users appropriate software and sometimes innovate. Some appropriations are simple and perform useful functions for the individual, such as setting an alarm (Chapter 5, Page 114) or automatically adding a signature line. Others affect a particular group, such as when a translator created a special set of customization files for the members of the video group, who all began using those files at the same time (Chapter 6, Page 6.2.1). Still other appropriations can affect the functioning of the software and its use throughout the organization, such as the effect of the Zephyr help list on the consulting organization (Chapter 5, Page 116) and the people who use consulting services. People may observe their own behavior and customize to encode commonly-repeated patterns (Chapter 5, Page 112). When shared, these customizations serve to create informal group norms and continued to affect the behavior of individuals within these groups for years to come.

8.1.6 What are the implications for organizational theory?

This research illustrates the importance of considering not only the consequences of individual interactions with technology, but also the effects of sharing of technology among members of an organization. Decisions made by individuals may affect the use of the technology by the organization, especially if those decisions are embodied in exchangeable files that continue to have an impact on user behavior over time. Software does not remain static when it is introduced into an organization. People in the organization evolve their individual patterns of use, share them with each other, react to external changes, both technical and non-technical, and sometimes proactively modify the system to produce real innovations.

The sharing of customizations is a subset of a larger set of issues, including how users learn to use new technology and the ways they innovate and appropriate the technology for their own purposes. As users learn a new technology, they are influenced both by its design and the established patterns of use by other members of the organization. At the same time, individuals innovate. This study has demonstrated that customization is not a purely individual activity. Members of an organization may become translators for others and create customization files for others to use. These files continue to affect behavior for long periods of times (often several years) and serve to establish and perpetuate standard patterns of behavior throughout the organization.

The triggers and barriers to customizing software appear similar to those for learning new features or innovating (Chapter 5, Page 105). While there are tremendous individual differences, if we understand the characteristics of the triggers and barriers to each of these activities, it may be possible to shift the trade-offs so as to increase both learning and innovation. This research argues in favor of the inclusion of micro-social analyses in order to more fully understand behavior at the macro level.

8.2 Practical questions

The second set of questions address more practical issues about how people use customizable software and how it affects their behavior in the organization.

8.2.1 Which users customize and under what conditions?

End users in both software environments customize their software, sometimes extensively. The range in customization behavior is wide, from no customization to a level that can exceed the time spent on actual work. Users are most likely to customize when they join an organization, which is also when they know the least about the technology and how they will use it over time. Unusual or purely aesthetic customizations are most likely to be added at this time, as users explore the new software. These kinds of customizations often break in subsequent software upgrades and are least likely to be reimplemented or fixed. Thus, many users experience a net loss of customizations over time, as they lose access to these early customizations and do not replace them with new ones.

One exception is the use of customization to encode learned patterns of behavior, such as aliases for commonly-typed items. Users continue to make these kinds of customizations at random intervals over time. The other exception are customizations that affect low-level skills, such as key bindings. Users create or borrow a set of key bindings when they first learn to use the system, which they rarely change. Both kinds of customization are likely to be retrofitted when the software is upgraded, to make the new version operate like the old. People who have either never customized or given up on customizing for some reason do not customize when the system is upgraded and instead learn the new patterns of interaction with the software.

Many people allow customizations to accumulate over the years. (Although a significant minority "clean them up" at random intervals or when they are looking at the customization file for some other reason.) The customizations that remain in active use are generally preserved when users move to a new system or experience system upgrades.

8.2.2 How do individual customizations affect group norms?

Customizations by individuals can affect group norms of behavior, and ultimately organizational norms, through sharing of customization files. Limited forms of sharing occurred in the Lens study by a technical 'wizard' and a less-technical staff member (Chapter 3, Page 60). More people shared customization files at Athena and two kinds of sharing, through two different mechanisms were apparent. Highly technical staff members broadcast or donate customization files by placing them in publicly accessible locations, sending announcements via Zephyr or electronic mail, or providing other staff members with access to their files (Chapter 6, Page 138). The last method of sharing takes advantage of an unusual characteristic of the Athena software environment. It provides an efficient and accessible way of allowing people to view or borrow working examples of other people’s customizations, without disturbing the owner.

The second sharing mechanism is supported by people who observe the needs of other users in the organization and translate the technology into terms that they can understand and use (Chapter 6, Page 144). These people are all less technical than the people in the first category, but are better at communicating with their peers and value their roles as people who serve the community. None of the managers provide formal support for these activities although several informally encourage sharing within their groups. Many managers are unaware that sharing goes on and at least one manager discourages it because she feels it interferes with the translator’s work. Most of the groups in this organization have a self-appointed translator, and A3 and E16 appear to serve this role for the entire organization (Chapter 6, Page 6.3.2). When a translator leaves a group, someone else usually steps in to fill the role. Similarly, if someone has been performing this role for some time and a new person arrives who volunteers to do it, the first person may stop and allow the second person to continue instead.

Users are most likely to borrow customization files when they first join the organization. Translators often help newcomers get set up, and provide them with a set of customizations that are designed to help them perform certain kinds of tasks. Some users in this environment then seek out the files of other, more experienced staff members or people who perform similar jobs. After a certain period of exploration, most users settle on a limited set of customizations and rarely change them. Users learn to use the system based on an informally gathered collection of customizations and usage patterns of other people in the group, plus the results of their own experimentation with the software.

Systems programmers, who are usually the first to try prototype or beta release software, are usually the ones who create the default customization files for the rest of the organization. Unlike the translators, these people are almost completely isolated from feedback by "regular" users and thus the files are often unnecessarily complex. These files become the basis for everyone else’s customizations, and the translators take on the task of creating working subsets of these files that make sense for different kinds of users on the staff.

These findings suggest that current activities by software manufacturers, managers and end users may work against the productive use of customizable software in organizations. The following sections suggest possible alternatives to current practices.

8.3 What are the implications for the software development process?

Software manufacturers, in response to customer demand, are increasingly making software applications customizable by end users. However, it is not clear that they consider the impact of design decisions on either individuals or on the results of sharing these customizations within an organization. The current software development process usually consists of a static ’needs analysis’ from which the functionality of the software is defined and the code implemented. Version 1.0 is usually released under tight time and budget restrictions. User interface concerns are often left "for the update" and the sets of default values that the first set of users may not be carefully chosen. Software development managers must make trade-offs about what to put in the first release and what to leave out. This research suggests that the trade-offs may be different when the use of the system by the end user is taken into consideration.

The first problem is that user needs are not static and it is difficult to identify them without actually talking to users. To limit the ’needs assessment’ to a short, one-time report that is examined only before the implementation begins is to risk building a great implementation of the wrong software. Software manufacturers are increasingly turning to usability testing as a way to obtain additional input from users during the development process. Unfortunately, this is still not sufficient. Such testing is usually done in a laboratory and users are brought in to work with a single application for a relatively short period of time. A variety of important issues may be evaluated and addressed in this way, but it is a mistake to believe that this will provide a real understanding of how users will use the software in their daily work. Behavior in a laboratory setting is not the same as that in a work environment. Short-term use may not be a good predictor of use over a long period of time. Users, including people who stay technically ’novices’, become used to the system, use it in conjunction with other software and off-line activities and develop a set of expectations about the technology, some of which change over time. Different interactions with the technology emerge over the days, weeks and months that people use the software and are influenced by how other people in the organization use the same technology. Thus, it is very important to add field-based user testing to the software design process, both to learn from user innovations and to see the patterns that people settle into. It’s also important to examine more than one site, because different sites have different software histories and norms of use, which may greatly affect the use of the new software.

Software manufacturers should also seriously 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. Users may react in different ways. Many of the users in the Athena 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. Some avoided using the software altogether. Others learn to use each new set of defaults, particularly those who feel they should use the 'standard' version of the software. These people are either greatly disrupted by changes in subsequent releases (and blame the software), or they begin to retrofit the software and continue to use the earlier sets of defaults for months or years. Unless users can evaluate their initial choices and create customizations that are likely to be useful for a long time, subsequent improvements by the manufacturer in the user interface are unlikely to be used.

Note that the solution is *not* to release software without any examples of customization. This simply shifts the burden of creating the first set of standards to an end user in the customer's organization, who may lack both technical skills and an appreciation of the elements of a good user interface. These people are unlikely to have either the technical skills or the time to systematically test the possibilities and design a useful starting point for future customizations. The "lead-users" in the organization, who try the new software first, may discard inefficient customizations created during their first interactions with the software, since they are more likely to be reflective about their own use of customizations and will make subsequent customizations accordingly. However, the people who receive early customizations from these lead users *are* likely to continue using these files, often for long periods of time. These people are also the ones most likely to 'help' newcomers by sharing these files, which will perpetuate any errors and inefficiencies in the original customization files.

Because users often appropriate the technology for their own use, software upgrades accommodate these appropriations into account or be rejected. Tatar (1990) has already demonstrated that when a technology is introduced to replace a previously manual operation, it often becomes evident that the manual operation was serving more than just the publicly-stated functions for which it was designed. In the same way, technology that has been in use in an organization is likely to have been changed as a result of that use, and characteristics that were not evident when the software was originally designed and introduced may become critical and only evident when a new "improved" program is created to take its place.

This research adds credence to the value of an iterative design process that actively includes users, even after the product is shipped. Users provide new insights about the software and may provide useful innovations as well. Many of these innovations stem from customizations that operate along dimensions not directly provided for in the original software design. Manufacturers who identify such innovations and incorporate them into subsequent versions of the software will both help the groups that identify them in the first place and also improve the overall usefulness of the software to a wider range of customers.

In summary, software manufacturers need to actively include users into both the software design and the software development process.[29]Software is not 'done' when version 1.0 is released, with subsequent upgrades as simply added features or bug fixes. The concept of the software evolves over time, as it is used. Software manufacturers should modify the software development process to include the dynamic relationship between users and the software and include them in all phases of the design, development, delivery and upgrade cycle. In fact, users provide opportunities for manufacturers to obtain deeper and different understanding of their software and may discover important innovations that will enhance future versions of the software.

8.4 What are the implications for software design?

Just as the software development process should take the dynamic relationship between users and the software into account, the actual features of the software should better support these activities by users. The following are suggestions for designing software that helps users adapt the software to meet their own needs.

1. Reflection: Provide users with feedback about the effectiveness of their customization decisions and the opportunity to reflect upon their actual processes of use.
2. Context: Allow users to encode patterns of behavior and informally include information about their current work contexts.
3. Sharing: Provide mechanisms for users to exchange software, especially among translators, to encourage innovation and share effective methods of accomplishing tasks.

[29]As a manager, I became a regular user of a particular spreadsheet application. I was also familiar with several other spreadsheets by other manufacturers. I discovered several bugs and had some suggestions, so I called the project leader. She told me firmly that no new ideas could be incorporated, because they had already determined their schedule for the next three years. Clearly, no additional functionality could be added. I asked her whether or not anyone in her group ever actually used spreadsheets. She said no, of course not. They were all programmers. Not surprisingly, this software application is years behind its competitors.

4. Dimensions: Provide dimensions of customization that match the user's tasks.

8.4.1 Development of reflective software

Individuals make choices about when to customize, responding to external or internal triggers or reacting to barriers that are too high. However, choices about whether or not to customize are usually based on perceptions, not actual behavior. Software that provides users with feedback about their actual use patterns can help users identify productive (and unproductive) actions, which should help users make choices that increase productivity.

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 how he or she actually uses 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 the 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 customizations.

8.4.2 Help users capture and customize work context

Customization options are generally presented as a set of features that can be modified individually. The user must provide all of the information about the current work context, to the extent that it affects decisions about which features to use. One of the most important roles of a translator is to define how these different sets of features fit together for specific work situations for an individual user.

In both the Athena and the Lens studies, some people created multiple versions of certain customizations which 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. In the Lens study, providing multiple rulesets, e.g. one for returning from vacation and another for daily use, proved extremely popular. In the Athena study, users created different window layouts when performing different jobs or for use on different machines.

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. Software manufacturers 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. Many software applications provide the concept of a "macro", which allows end users to demonstrate or define a set of actions that can be run at a later time. Particularly in the Athena study, users were most likely to say that encoding a commonly-repeated pattern of behavior was useful. 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 this study invented a number of different ways of encoding these patterns and used them extensively.

8.4.3 Provide support for sharing customizations

Specific decisions about the design of customization features have significant effects on how the software is used. For example, providing customization capabilities solely via a "direct manipulation" interface may make it easier to modify individual features but harder to share, because users might not understand the form in which the customizations are stored nor be able to gain access to them. Software manufacturers should also 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 become the translators or sources of 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.

8.4.4 Determine effective dimensions for customization

Software can be designed to be very flexible, but not infinitely so. As pointed out in Chapter 1, providing some structure may actually provide more opportunities for innovation, because users have a conceptual framework from which to work. The difficult problem is to determine what the most useful frameworks are, and what the corresponding dimensions of customization should be. For example, in a spreadsheet, it is possible to display the numbers contained in the grid in a variety of ways. However, it is not easy to make the grid three-dimensional instead of two-dimensional. The grid of rows and columns is a primative for a spreadsheet. Making changes across primitive boundaries is difficult; making changes within the boundaries should be relatively easy.

Note, however, that it does not always make sense to expose the internal structure of the software to the user, as part of the user interface. This is a long-standing argument, between the use of 'black box' designs that hide the internal structure and 'glass box' designs that expose the structure. However, the software designers must recognize that when they make decisions about the underlying structure, they are making decisions that affect the dimensions along which the software can be made to be customizable.

The Lens study provided an example in which the original software design did not prevent a different dimension of customization, but did not make it easily accessible either. The new version of the software provided support for multiple rulesets, which provided a new dimension along which people could customize. Although this did not fundamentally change the architecture of Lens, it did require a programming change in the code, rather than just a customization, to add the new customization dimension.

Different customization mechanisms may have a range of positive or negative consequences when used in a real work environment. Tools that allow users to observe their own processes of use to improve personal satisfaction and productivity may also be used by managers to evaluate performance. If used poorly, they may lower user satisfaction and create an environment of the sort that Zuboff warns against. Similarly, tools that permit sharing do not affect the quality of what is shared. People may be better off *not* borrowing certain kinds of files in certain circumstances. The trick is to create software that encourages positive kinds of use and does not interfere with the informal communication mechanisms that members of the organization develop on their own. This has implications for managers, who can affect the ways in which software is used in the organization.

8.5 What are the implications for managers?

Studies of white collar productivity indicate that the introduction of computers has been correlated with a *decrease* in productivity (Dertouzos et al, 1989). Although computers offer flexibility and power, productivity gains will only be achieved if users *use* the technology effectively. The results of this study indicate that the current mechanisms for encoding patterns of use often work against innovation and may even decrease productivity over time. Users actively resist change by customizing their software to be like previous systems, even if those systems are now obsolete. Users tend to customize when they are least able to make optimal judgements, as when they are just starting to use the software and understand the least about both the software and their own use of it. They will avoid customizing or learning more about the system unless forced to by external events.

Managers can provide guidelines and support to help people in the organization give and receive help more effectively.[30] The guidelines should include information for both givers and receivers and make sure that there are some feedback loops, to make sure that the help is effective. Many organizations provide their employees with training about the software they use. Teaching often occurs in a classroom setting and is generally provided by the software vendor or an internal training group. Some organizations provide "advanced courses", sometimes immediately after the initial training period and sometimes later. Some organizations establish user groups, either formally or informally, to exchange information about the use of the software.

Training is often provided when an employee first joins the organization, before he or she fully understands either the job or the technology. Training may also be necessary when a new software product is introduced into the organization. Such training is often considered a "perq" particularly in organizations that provide users with some latitude in their use of the software. Because training is expensive, especially when provided by outside vendors, organizations often limit the number of people who attend. These people are then expected to transfer their knowledge to others in the organization. These data suggest that it would be most beneficial for organizations to choose people who understand the needs of their fellow workers and can translate the new technology into terms those people understand. However, because access to training is often based on technical expertise or treated as a reward, people with the best people skills may not be the ones who go.

Even technical experts face problems when learning about new software. They do not yet understand the details of how the software will be used in the context of their daily work and the patterns of use they establish will be influenced by their own preferences, past experience with other software and the way the software is presented by the teacher. Training that is provided by outside vendors may present additional problems. The courses may serve a marketing function for the software manufacturer and teachers may concentrate on the new features that distinguish the software from the competition. It is unlikely that such training will be optimized for a particular organization’s needs. The course attendee’s initial experiences with the software will form the basis for the information that will be shared with their colleagues. Although these individuals may continue to experiment over time and increase their own efficiency, these data suggest that their colleagues will not. Once established, usage patterns are difficult to break and users will actively try

[30] I always wanted to write a book called "How to get help from a hacker", for non-technical people who need to get help from people who are more interested in the technology than in how it can help solve the end user’s needs. It should help people deal with the mismatch between a hacker’s view of the system and the end user’s use of it.

to maintain them in spite of changes in the software environment. The "standard" patterns of use that are thus established are suboptimal.

Thus, "standard" patterns of use are generally established by people at the time in which they understand the technology the least and have had the least experience working with it in their daily jobs. The people who are most likely to bring new technology into an organization are not usually the people with the most people skills and will pass on their first, suboptimal attempts, which will be used by less technical members of the organization for long periods of time.

One of the problems here is that users become increasingly reluctant to change their patterns of behavior and in fact work to preserve them in the face of change. This provides short-term benefits for the user, who can get the next job done quickly. But if people spend time customizing new software to be like older, more obsolete software, it will have a chilling effect on long-term productivity. Users make a short-term investment to avoid learning something new, rather than a longer-term investment that will help them in the future.

One way to address the problem is to provide users with opportunities to reflect upon their use of the software using real data about the effectiveness of their use of the software. This involves allowing users time away from work (similar to taking a class) in which the topic of individual productivity is addressed. Different techniques will work differently for different individuals. A simple example is the difference between touch-typists and non-touch typists. The former may benefit from learning additional keyboard commands and the latter may benefit from learning how to customize menus to make them more optimal.

An important implication is that managers will benefit most by helping translators become more effective. Much of their behavior is done without management support or rewards or official mechanisms. Highly informal behavior by these individuals can have a significant impact on the overall use of the technology, either positive or negative. Although people may learn to avoid help from certain people over time, most of the damage is done when people first arrive, before they learn who to ask and who not to ask. In these studies, translators appeared to perform their role only when they perceived the need. If managers reward effective translators for performing such services, other people will be less likely to act in this role.

Just as managers support periodic maintenance of hardware, they should also support periodic maintenance of user skills. Managers should consider the following:

1. Skill maintenance: Bring groups of people together to reevaluate their use of the software on a regular basis, perhaps once a year.

2. Share files: Encourage people to bring customization files (or useful tricks they've learned) to share with others. Useful techniques may be exchanged and maintained, less useful techniques should be identified and allowed to die out.

3. Reflection: Allow time for individuals to reflect upon their own use of the software and how others use the software.

8.6 What are the implications for end users?

These data indicate that users do not, in fact, optimize their use of the software. End users of customizable software should recognize that their early customization decisions may not reflect their eventual use of the software. Similarly, while it is useful to obtain working examples of files from others, particularly from people acting as translators, it is important to periodically evaluate the effectiveness and the quality of the customization files. Sometimes this can be done by simply asking for advice from another person who uses the system in a similar way.

End users often reach situations in which a particular kind of customization is deemed desirable. The user must evaluate the short-term costs of making the customization, including the possibility of making a time-consuming error or taking a significant among of time and not accomplishing either the task or the customization. This must be weighed against an unknown, long-term benefit. It is little wonder that end users do not customize as much as they say they would like and avoid making a few customizations that might significantly enhance productivity. As a user, it makes sense to periodically set aside a short amount of time, perhaps in the form of a meeting with someone who knows the system, and explicitly think about current use of the system and needs. If it is possible, the user should try to get feedback about actual use of the system, because user's perceptions are often wrong.

8.7 Limitations of this research

This research uses an ethnographic style of analysis, with inherent benefits and limitations. Although the problems are bounded, the research still provides value within its scope. This research answers some questions and poses more; it suggests caution for software manufacturers and managers who make and use customizable software, it highlights phenomena that traditional research models and methodologies miss and suggests directions for future research. This section addresses several limitations of this research.

8.7.1 Relevance of studying these groups

Ethnographic research is commonly questioned about the generalizability of the results. Such studies provide an in-depth analysis of a small group of people rather than a less-detailed analysis of a very large group of people. The concern is relevant if the goal of the research is to make generalizations about average behavior in the large group based on data from the small group. However, that is not the intent of this research. One of the important assumptions underlying this work is that context is an important component in human interaction with technology. Thus, much of the behavior that we see is context-dependent and may differ greatly from one organization to the next. Instead of concentrating only on the 'average' or most common behavior and discarding the rest as 'noise' or random variation, this style of research explicitly looks for variation and examines it. However, this is not to say that generalizations cannot exist at all. Certain patterns of behavior may be detected, such as the presence of translators, which may occur in other organizations. If so, future studies may choose to investigate this particular pattern in greater detail or attempt to determine which kinds of environments encourage these patterns.

Another question is why study these particular organizations? In some sense, each is quite different from a 'typical' business organization (if such a thing exists). Both have created software that has influenced the entire software industry, particularly at the user interface level. This software has resulted in multi-million dollar investments by other corporations and has influenced or created internationally-recognized software standards.[31]

These organizations are both lead users of leading edge technology. As the first active users, they provide the first opportunity to study the effectiveness of the technology in the field as opposed to the laboratory. Studying them can help us understand how people will react to technology that will be in wide-spread use in five to ten years.

Another important issue is the relationship between these organizations and innovation. Both organizations value and have been successful at it. Staff members often try new software and some (although not all) groups within both organizations actively value innovation and strive to create novel and effective uses of the technology. Each have on-site technical experts who can

[31]For example, the X Window System created at Athena, was made into an international software standard in record time. The Open Software Foundation was started as a $100 million joint investment between Digital, Hewlett Packard, IBM and other major computer vendors to develop and extend the X Window System. Even its competitors, Sun and AT&T, support it. Digital has invested over a billion dollars in the DEC version of the X Window system, which is by far the largest software effort ever undertaken at Digital and perhaps in any company. Even though it did not exist four years ago, the X Window System is now available on wide variety of workstations and personal computers.

incorporate changes based on feedback from users. To the extent that these organizations use the software in innovative ways, it is useful to study what they do. To the extent that they are *not* innovative, it is important to begin to understand why. For example, if staff members at Project Athena have problems with the X Window System, then it is quite likely that organizations without the resources of Project Athena are likely to also experience problems.

Although these groups have a number of unique characteristics that make them interesting to study, they also share many characteristics with other organizations. Both employ a large number of people with jobs that are similar to those in any corporation, including secretaries, non-technical managers, financial administrators, etc. The majority of the staff in both organizations do not program for a living. The staff of Project Athena are similar to the staff of an MIS department in a large corporation, with a number of programmers and a majority of non-programmers who handle administration. The Athena staff provide a service for over 10,000 demanding users and face very real deadlines. They must make the kinds of decisions and trade-offs about the use of the technology that members of any service organization must make. Thus many of the patterns detected here may apply to people in other organizations as well.

8.7.2 Rate of technological change

Another issue is the rate of technological change experienced by the members of staff. Both organizations experience higher than normal rates of change, although the changes are controlled and software releases are announced and usually predictable. In the Lens study, major releases occur every four to eight months. At Athena, major releases occur three times a year, at the beginning of each semester. In organizations with centrally-managed time-sharing systems, such change may occur once a year or once every two years. The rate of change may be faster or slower for people with personal computers, depending on the individuals who make purchasing decisions and the attitudes about using free 'shareware' versus fully supported professionally-developed software applications. In some cases, users may maintain the same software environment for three or four years or perhaps more. However, change does occur. The reactions of the Athena staff, particularly those who are not programmers, may represent one form of reaction to such change. (This group is likely to react *better* to change than other organizations, given the technical talent locally available, however, they still actively resist change.)

8.7.3 Limitations of the data collected

Although I have been able to collect a variety of different kinds of data, there are additional forms that would be useful as well. For example, in the Lens study, I had access to a complete record of all changes in Lens rules for over two years. In the Athena study, the data was collected in the form of snapshots that provide a complete record of change for people who do not change files often and a potentially incomplete record for people who change their files often. This is because snapshots are collected on a particular date. The date of the most recent change and the date of the change before that are both recorded. If more than two changes occur between two snapshots, the dates of intermediate changes to files are lost. I have partially addressed this problem by taking additional snapshots for people with higher rates of change.

Another useful type of data would be a record of the dates and transcripts of the communications about customization among staff members, especially by translators. However, that is clearly beyond the scope of this study. The important result here is that the translators exist and have an effect on the organization. A longitudinal study that measures both on-line and off-line communication would be necessary to accurately trace the influence of these people on the organization over a period of time.

Another difficulty is that users forget their "micro-level" decisions about when to customize. Thus, in the example of my own customization use at the beginning of Chapter 5, I might not have remembered the problem or why I didn't fix it, if I hadn't just experienced the phenomenon. Thus, it is important to recognize that many decisions about when to customize, adjust, learn, or innovate are made fleetingly, on the spur of the moment, very much influenced by the immediate factors in the users environment. This matches Suchman's concept of "situated action", in which people may derive a post-hoc analysis of what they do, but that in the actual course of action, their behavior is much more affected by the immediate environment around them. The use of a variation of the critical incident technique, in which the participant recounts a recent important incident, helps users to remember more detail, but not all. Thus, it is important to recognize that these data underrepresent the factors that influence users making customization decisions. Note that this is a situation in which certain kinds of laboratory studies might shed additional light on the situation, especially if the users can bring actual work into the laboratory setting. Given the richness of the data and the range of possible issues to study, the interviews provide valuable insights about promising phenomena and patterns to study further.

8.7.4 Co-adaptive phenomena

Co-adaptive phenomena may occur in many places, under many conditions. It is not my intent to question whether or not co-adaptive phenomena exist. With a little reflection, one can think of numerous examples.[32] What is important is whether or not the characteristics of co-adaptive phenomena are present in the interaction between people and technology and whether or not this view provides insights and causes us to adjust our theories and research methodologies to accommodate it.

In some situations, the effect of the technology on the users may overwhelm the effect of the users on the use of technology. In other cases, the reverse may be true. To the extent that those are the phenomena of interest, it may not be necessary to take the co-adaptive perspective into account. However, I contend that many studies that do not take this into account are missing an important aspect of the behavior. This work is particularly relevant in such situations.

8.8 Directions for future research

This research poses a number of questions that can only be addressed with additional research. This section suggests some of the possible avenues that such research might take. Some of these research questions require additional methods and theory.

8.8.1 In-depth examination of translators

Translators play a very important role, particularly in the Athena environment. Sharing of customization files provides an informal mechanism for cooperative-work. Several recent studies find similar sharing with spreadsheets (Nardi and Miller, 1990) and Notecards (Trigg et al., 1987), and the translators at Athena share some characteristics, though not all, with the gatekeepers identified by Allen (1972). Additional research is necessary to understand how people choose to serve as translators (and how they decide when to stop serving that role). How aware are managers of this function and do their activities encourage or discourage people? In the Athena study, one person was essentially removed from his role as a translator by a concerned manager. Yet another person who generates some of the same kinds of problems continues to perform the

[32] I had a discussion with Vik Vyssotsky, Director of Digital's Cambridge Research Lab, who immediately came up with an example. He described an example of the process he goes through when buying a new bicycle. He first finds one that is basically the right size with the appropriate features. Then, he begins to ride it. He realizes that several things are not adjusted quite right, makes those changes, and rides again. Over time, the adjustments become smaller and eventually, the bicycle is adjusted properly. If someone borrows it, the adjustments must be made again.

same function without his manager's knowledge of the consequences. How is this behavior regulated, if at all? Is it all ad hoc or do some organizations develop more formal strategies for evaluating the effectiveness of shared customizations and encourage only certain kinds of sharing?

8.8.2 Customization in other environments

These studies examine two mechanisms of customization. In the Lens study, the whole purpose of the software is to provide users with the ability to customize their processing of electronic mail. In the Athena study, customization was provided through specific customization files, some of which affected a single application and others (particularly the window manager) which affected groups of applications. These two methods hardly exhaust the possibilities. Some software is being retrofitted to include new customization features, which are made accessible through a variety of mechanisms. Other software is being designed from the beginning to be highly customizable (e.g. Notecards, Hypercard and emacs) and they too use a variety of mechanisms for providing users with the ability to customize? Which mechanisms are most effective in which contexts? How much are customization patterns in individuals affected by the different characteristics of the software and the different characteristics of the organization? How do different decisions about customization features affect the sharing of customizations (and their quality) within an organization? How do different rates of technological change affect customization behavior?

These studies suggested that there are different kinds of customization and that the rates of customization respond differentially over time. Customizations generated as part of the initial exploration of the system were not retained when system upgrades causes problems with them, whereas changes that encoded low-level skills, such as key bindings, or commonly-repeated patterns, were adjusted with each new change. Is this common? Are there other kinds of customizations that users modify differentially over time? Do different customization mechanisms affect the changes users make over time?

8.8.3 The role of context

How important are changes in the user's work context with respect to the effective use of the technology? These studies identified situations in which users innovated by incorporating information about the external work context into particular sets of customizations (without explicitly articulating the context). Is this an unusual circumstance? Or is it a new dimension of customization that will enhance user's ability to work in a changing environment? Can we explicitly accommodate user context as a way of changing the user's interaction with the software? Should the ability to incorporate user context be a component in the evaluation of user interfaces and customization capabilities? Additional work is necessary to understand both how users perceive this notion of context and how it can be incorporated most effectively.

8.8.4 What are the effects of external events on customization?

In both of these studies, users responded to certain kinds of external events by modifying their rules or their customization files. It appeared that the events most likely to cause change were those that caused users to reflect upon their use of the software and provided the time to actually modify the files as a result. Is reflection really the important factor here, or is this just a variation of the Hawthorne effect? This question could be studied directly with a controlled field study, examining the effects of different kinds of external changes on the rate and effectiveness of customization changes. (Note: the study should include the effect of the user's own changes and how they affect future changes.) Such a study would help managers understand the effects of changes on processes over which they have little or no control.

8.8.5 Relationship between customization, learning and innovation

I have argued that user's decisions about customization share some characteristics with general learning about the technology and with the creation of innovative uses of technology. However, these data only provide indirect evidence for this claim. Are the triggers and barriers identified here similar to those for choices about learning and innovation? If not, how do they differ? Do people in the organization use the same informal mechanisms for sharing learning and new ideas as they do for sharing customizations? Does the presence of a customization 'artifact' have a differential impact on both future decisions and on sharing? Research into these questions will require the development of methods for unobtrusively recording learning and identifying innovations, and tracing how they move through the organization.

Another issue is the effect of this kind of environment on customization, learning and especially innovation. The sites of these two studies are known for their high levels of innovation, although not necessarily among the individuals studied. Do other organizations that place a greater emphasis on production and less on innovation, exhibit similar levels of customization and sharing, or do other patterns emerge? If these two sites are more innovative, what are the characteristics that make them so? Can any of these characteristics be transferred to environments with a greater emphasis on production?

If studies are conducted to show that reflective software increases the effectiveness of customization, is there a similar effect on learning?

8.9 Summary of the contributions of this research

In this dissertation, I have examined customization behavior in two sites, using two different forms of customization. I have used the perspective afforded by the structurational model (Orlikowski 1989) to examine a variety of influences on the behavior. In addition, I have demonstrated that customization behavior is a co-adaptive phenomenon. I have added a micro-level of analysis surrounding the customization decisions of individual users, which both enriches the structurational model and explains one of the mechanisms by which informal sharing by individuals in the organization can affect the norms of behavior sanctioned by the organization. The following is a summary of the specific findings of this dissertation.

1. The specific identification of co-adaptive phenomena, supported by field data.
2. The theoretical linking of co-adaptive phenomena and the structurational model, providing evidence of a mechanism by which individual interaction with technology affects organizations.
3. The discovery of specific patterns of customization:
    a. Users are most likely to customize when they join an organization, when they know the least about the technology and their use of it.
    b. Customization provides a useful way to explore a new software environment.
    c. On-going customizations most often involve encoding commonly repeated patterns of behavior, rather than adjusting static features.
    d. System upgrades cause users who have already customized software to retrofit the software to be like the previously-learned software.
    e. Over time, most users customize less, except to encode common patterns and maintain stability. This is true regardless of level of technical expertise.
4. The discovery of specific patterns of sharing:
    a. Users are most likely to borrow customization files when they join the organization.
    b. Translators interpret individual users needs and create specific customizations organized to meet those needs.
    c. System programmers create the first versions of customizations without feedback, which then become the basis for everyone else’s customizations.
5. The proposal for changes in the software development process, including use in the field as an important input to future development.
6. The proposal for software manufacturers to include mechanisms that explicitly support reflection about use of the software and for sharing of customizations.
7. The proposal for managers to support periodic "maintenance" of skills to allow users to reflect on and improve their productivity and to share innovations.

References

Abelson, H. and Sussman, G. (1985). *The Structure and Interpretation of Computer Programs*. Cambridge, Massachusetts: The MIT Press.

Ackerman, M. and Mackay, W.E. (May 1989). Context Issues for Users in Multi-Windowing, Multi-Tasking Environments: The Case of X. *CHI '89 Workshop on Context: in the eyes of users and in computer systems*. Austin, Texas: ACM/SIGCHI, Position Paper.

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

Barley, S. R. (1990). The alignment of technology and structure through roles and networks. *Administrative Science Quarterly*, *35*, 61-103.

Barley, S. R., Meyer, G.W., Gash, D.C. (1988). Cultures of Culture: Academics, Practitioners and the Pragmatics of Normative Control. *Administrative Science Quarterly*, *33*, 24-60.

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.

Blau, P., McHugh-Falbe, D. McKinley, W. and Phelps, T. (1976). Technology and Organization in Manufacturing. *Administrative Science Quarterly*, *21*, 20-40.

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.

Brown, J. and Van Lehn, K. (1980). Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science*, *4*, 379-426.

Burton, R.R. & Brown, J.S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies*, *11*, 5-24.

Card, S. and Henderson, A. (April 1987). A multiple, virtual-workspace interface to support user task switching. *CHI + GI Conference Proceedings*. Toronto, Canada.

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

Champine, G. (1989). *Project Athena as a Distributed System* (Technical Report). Digital Equipment Corporation.

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

Cohen, K. (November 1987). Effects of Campus Computerization: Evaluation Efforts and Findings. *IBM ACIS University Conference Proceedings*. ACIS.

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*. Portland, Oregon.

Feldman, M. (December 1986). Constraints on Communication and Electronic Mail. *Proceedings on the Conference for Computer-Supported Cooperative Work*. Austin, Texas.

Fischer, G. and Girgensohn, A. (April 1990). End-user modifiability in design environments. *CHI '90 Conference on Human-Computer Interaction*. Seattle, Washington: ACM/SIGCHI.

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

Gorry, G., Burger, A., Chaney, R., Long, K., Tausk, C.M. . (September 1988). Computer Support for Biomedical Work Groups. *Proceedings on the Conference for Computer-Supported Cooperative Work*. Portland, Oregon.

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

Henderson, A. (1990). Tailorability. In Kyng, M. (Ed.), *???*. : .

Jackson, G. (1988). Settling Down with Athena. Internal Report, MIT Project Athena.

Kling, R. (1980). Social Analyses of Computing: Theoretical Perspectives in Recent Empirical Research. *Computing Surveys*, , pp. 61-110.

Kling, R. and Iacono, S. (1984). Computing as an Occasion for Social Control. *Journal of Social Issues*, , pp. 77-96.

Kuhn, T.S. (1970). *The Structure of Scientific Revolutions*. Chicago, Illinois: University of Chicago Press.

Lai, K. and Malone, T. (September 1988). Object Lens: A Spreadsheet for Cooperative Work. *Conference on Computer-Supported Cooperative Work*. Portland, Oregon: ACM.

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

Latour, B. (1987). *Science in Action*. Cambridge, Massachusetts: Harvard University Press.

Levitt, and Whistler. (1958). . *Harvard Business Review*, (), .

Liddle, D. (1990). What makes a desktop different? Alsop, S. (Ed.), *The Agenda 90 Proceedings*. Carlsbad, California.

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

Mackay, W.E. (August 1986). Lens Reference Manual and User's Guide. Massachusetts Institute of Technology.

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

Mackay, W.E. (1988). *A Designer's View of the X Window System*. Atlanta, Georgia: ACM/SIGGRAPH. Tutorial Notes Published at SIGGRAPH '88 Conference.

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., Gardner, B.R., Mintz, T.H., Pito, R.A., and Siegal, J.B. (August 1989). Argus Design Specification. Distributed by the Open Software Foundation and MIT Office of Technology Licensing.

Mackay, W.E., Malone, T.W., Crowston, K. Rao, R., Rosenblitt, D., and Card, S. (May 1989). How do experienced information lens users use rules? *CHI '89 Conference on Human-Computer Interaction*. Austin, Texas: ACM/SIGCHI.

MacLean, A. Carter, K. Lovstrand, L. and Moran. (April 1990). User-tailorable systems: Pressing the issues with buttons. *CHI '90 Conference on Human-Computer Interaction*. Seattle, Washington: ACM/SIGCHI.

Malone, T.W. (January 1983). How Do People Organize Their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Information Systems*, *1*(1), 99-112.

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.

Malone, T.W., Grant, K.R., Lai, K.Y., Rao, R. & Rosenblitt, D.R. (1987). Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination. *ACM Transactions on Office Information Systems*, *5*(2), 115-131.

Markus, L. and Robey, D. (1988). Information Technology and Organizational Change: Causal Structure in Theory and Research. *Management Science*, *34*(5), 583-598.

Mintzberg, H., Raisinghani, D. and Theoret, A. (1976). The Structure of 'Unstructured' Decision Processes. *Administrative Science Quarterly*, *21*, 246-275.

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.

NCAR. (1985). *National Center for Atmospheric Research Annual Report* (Technical Report). Boulder, Colorado: University Corporation for Atmospheric Research.

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

Noble. (1984). *Forces of Production: A Social History of Industrial Automation*. New York, New York: Oxford University Press.

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

Norman, D.A. and Draper, S.W. (1986). *User-Centered System Design*. Hillsdale, New Jersey: Ehrlbaum Associates.

Orcutt, R. (1990). . Project Athena Memo.

Orlikowski, W. (1989). The Duality of Technology: Rethinking the concept of technology in organizations. Unpublished manuscript.

Orlikowski, W. and Baroudi, J.J. (1989). *IS Research Paradigms: Method Versus Substance* (Technical Report). Cambridge, Massachusetts: Sloan School of Management, Massachusetts Institute of Technology. Sloan Working Paper # 3028-89-MS.

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

Scheiffler, R. & Gettys, J. (1986). The X Window System. *ACM Transactions on Graphics*, *63*, 2-29.

Sidman, M. (1960). *Tactics of Scientific Resarch: Evaluating Experimental Data in Psychology*. New York, New York: Basic Books.

Sondheimer, N. (198?). Resistance of Programmers to Using New Features. *Find reference. Bell Labs, Illinois.*, , pp. .

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

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.

Steiner, J., Neuman, R. and Schiller, J. (February 1988). Kerberos: An Authentication Service for Open Network Systems. *USEnix Conference Proceedings*. Dallas, Texas.

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

Tatar, D., Foster, G. and Bobrow, D. (1990). Cognoter and conversation. , Vol. ().

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

Turbak, F. (1990). Personal communication.

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

Turkle et al. (1989). Project Athena. Internal Report, MIT Project Athena.

Van Praag, J. (1985). *A New Approach to English Language Learning by Computer by Means of Template Extraction from Wizard of Oz Transactions* (Technical Report 413). Software Human Engineering: Digital Equipment Corporation.

Von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science*, *32*(7), 791-805.

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.

Appendix A
Information Lens Questions

Interviews were conducted in the participant's office. Every participant was asked background questions about length of employment, job category, job requirements, programming background, etc. For each interview, each participant was asked to:

1. set up a one-hour appointment,
2. save messages from the past 24 hours (omitting confidential ones), and
3. if relevant, save the current Lens profile.

Different sets of interview questions were asked at different times. The general questions about mail use were asked at each interview. The open-ended questions about mail use were asked at the preliminary interview and some were asked in subsequent interviews.

1. **General questions about mail use**
   a. How many messages did you send today?
   b. How many messages did you receive today?
   c. Is this a typical day? Explain.
   d. How many mail folders do you have?
   e. How many messages are in your inbox?
   f. Is this typical? Explain.
   g. How many distribution lists do you subscribe to?
   h. How often do you read your mail?
   i. Do you read all of your mail?
   j. What percentage of messages do you wish you had never seen?
2. **Open-ended questions about mail use**
   a. Describe how you use mail.
   b. In what categories do you place your mail messages?
   c. Can you think of times in the past week in which you needed technical information? What did you do?
   d. Can you think of times in the past month when you've looked at a previously filed message? Describe the procedure you used to find it.
   e. For what types of communication do you prefer mail? Phone? Face-to-face conversation? Do you like electronic mail?
   f. In the past week, have you sent mail that you think would be of interest to people other than those you've explicitly addressed? Example?

g. In what categories do you place your incoming messages? What kinds of outgoing messages do you send?

h. Can you think of times in the past week in which you needed technical information? What did you do?

i. Can you think of times in the past month when you've looked at a previously filed message? Describe the procedure you used to find it.

3. **Questions about Lens use**

a. What kinds of rules would you like to process your mail?

b. Have you created any rules?

c. Were today's messages sorted correctly?

d. Do any rules delete messages?

e. Have you modified any rules?

f. Have you asked to modify any message types?

g. Have you ever addressed a message to "anyone"?)

h. Can you think of examples in which it would be better if most of your colleagues had access to Lens?

i. What would be the effect of not using Lens? Do you plan to use it after the study is over?

j. Can you think of any additional uses of Lens?

k. What influenced your decision to use Lens? Examples?

l. Has Lens changed the way in which you communicate with others? (prompt for: quantity, quality, type of message)

m. What do you like best about Lens? Least? Does Lens help you do anything you couldn't do before?

Appendix B
Athena Customization Questionnaire

Name:	____________________	Job title:__________________
Username:	____________________	Date:_______________________
Extension:	____________________	Office:_____________________
Here for:	months______________	years_______________________
Workstation:	public______________	private_____________________

Athena provides a variety of software and ways of customizing it. I am interested in which software you use and how you customize it, particularly through editing ".dot" files. I am also interested in other forms of customization and unique uses of software you may have developed.

1. Please print a copy of your workstation's version file. Type: `cat /etc/version | lpr`

2. Please print out a listing of your current "dot" files. Type: `ls -l -t -A ~/.??* | lpr`

3. Did you convert your Window Manager in this release?

   [ ] uwm -> mwm (yes)
   [ ] uwm -> uwm (no)
   [ ] haven't decided yet
   [ ] mwm -> mwm (no)

4. How much programming have you done? Years:_____________

   [ ] No programming courses
   [ ] 1 - 2 programming courses
   [ ] 3 - 5 programming courses
   [ ] Computer science minor
   [ ] Computer science major
   [ ] Computer science grad. school
   [ ] Never programmed
   [ ] Programmed high-level languages (hypercard)
   [ ] Programmed for own purposes, e.g. research
   [ ] Applications programmer job (years______)
   [ ] Systems programmer job (years______)

5. How many other kinds of computer systems have you used? ___________________

6. How often do you use your workstation?

   [ ] > 6 hours / day
   [ ] 2-6 hours / day
   [ ] < 2 hours / day
   [ ] 2 days / week
   [ ] 1 days / week
   [ ] < 1 day / week

7. Which of the following have you used to help you customize?

(Check all that apply.)	Use often	Use sometimes	Use rarely	Never use
Athena's documentation	[ ]	[ ]	[ ]	[ ]
Unix books	[ ]	[ ]	[ ]	[ ]
man pages	[ ]	[ ]	[ ]	[ ]
on-line browser	[ ]	[ ]	[ ]	[ ]
olc	[ ]	[ ]	[ ]	[ ]
ask someone	[ ]	[ ]	[ ]	[ ]
copy & experiment	[ ]	[ ]	[ ]	[ ]
write own copy	[ ]	[ ]	[ ]	[ ]
read the source	[ ]	[ ]	[ ]	[ ]
Other_________________	[ ]	[ ]	[ ]	[ ]

	Used how often?	Customized how much?	Customized how often?	Source of 'dot' or customization files?
	D = daily W = weekly M = monthly N = never used H = Halted use	P = programmed 2 = major edit 1 = minor edit 0 = never	D = daily W = weekly M = monthly (or explain)	I = I wrote C = I edited copy F = friend wrote A = auto copy only
Window manager				
uwm	[ ]	[ ]	[ ]	[ ] .uwmrc
mwm	[ ]	[ ]	[ ]	[ ] .mwmrc
awm	[ ]	[ ]	[ ]	[ ] .awmrc
	[ ]	[ ]	[ ]	[ ]
Text editors				
emacs	[ ]	[ ]	[ ]	[ ] .emacs
ez	[ ]	[ ]	[ ]	[ ] preferences
	[ ]	[ ]	[ ]	[ ]
Formatters				
Scribe	[ ]	[ ]	[ ]	[ ] scribe database
Tex	[ ]	[ ]	[ ]	[ ] Tex database
LaTex	[ ]	[ ]	[ ]	[ ] LaTex database
nroff	[ ]	[ ]	[ ]	[ ]
	[ ]	[ ]	[ ]	[ ]
Electronic mail				
mh	[ ]	[ ]	[ ]	[ ] .mh_profile
xmh	[ ]	[ ]	[ ]	[ ] .mh_profile
rmail	[ ]	[ ]	[ ]	[ ] .rmail
andrew	[ ]	[ ]	[ ]	[ ] .MESSAGES
	[ ]	[ ]	[ ]	[ ]
Communication				
zephyr	[ ]	[ ]	[ ]	[ ] .z files
discuss	[ ]	[ ]	[ ]	[ ] .meetings
NetNews	[ ]	[ ]	[ ]	[ ] .newsrc
OLC	[ ]	[ ]	[ ]	[ ]
	[ ]	[ ]	[ ]	[ ]
Languages				
C	[ ]	[ ]	[ ]	[ ] make files
LISP	[ ]	[ ]	[ ]	[ ]
FORTRAN	[ ]	[ ]	[ ]	[ ]
	[ ]	[ ]	[ ]	[ ]
Unix				
C shell	[ ]	[ ]	[ ]	[ ] .cshrc.mine
awk	[ ]	[ ]	[ ]	[ ] .Xresources
yacc	[ ]	[ ]	[ ]	[ ] .Xsession
	[ ]	[ ]	[ ]	[ ] .startup.X
				[ ] .path
				[ ] .environment
				[ ] .logout
Applications				
rs1	[ ]	[ ]	[ ]	[ ]
Matlab	[ ]	[ ]	[ ]	[ ]
2020	[ ]	[ ]	[ ]	[ ]
idraw	[ ]	[ ]	[ ]	[ ]
Argus	[ ]	[ ]	[ ]	[ ]
Muse	[ ]	[ ]	[ ]	[ ]
	[ ]	[ ]	[ ]	[ ]

Appendix C
Athena Communication Network Questionnaire

Note: Names have been removed for reasons of confidentiality. The actual questionnaire has two pages and includes names from the current staff list, as well as names of people who have left Project Athena.

Name:____

Username:_________________________ Date:_______________ Page______of_______

I am conducting a study of how people customize their software and the ways in which those customizations are shared within a group. Your participation in this study is voluntary. If you decide to participate, it will take 10-20 minutes to fill out the following questionnaire.

1. Please check how often you speak to the following people (to help you remember, I've listed the names are from a recent staff list.) Please add a comment if one of the categories does not fit.

Name	Username	Daily?	Weekly?	Monthly?	Never?	Comment?
a	a	[ ]	[ ]	[ ]	[ ]	_____________________
b	b	[ ]	[ ]	[ ]	[ ]	_____________________
c	c	[ ]	[ ]	[ ]	[ ]	_____________________
d	d	[ ]	[ ]	[ ]	[ ]	_____________________
e	e	[ ]	[ ]	[ ]	[ ]	_____________________
f	f	[ ]	[ ]	[ ]	[ ]	_____________________
g	g	[ ]	[ ]	[ ]	[ ]	_____________________
h	h	[ ]	[ ]	[ ]	[ ]	_____________________
i	i	[ ]	[ ]	[ ]	[ ]	_____________________
j	j	[ ]	[ ]	[ ]	[ ]	_____________________
k	k	[ ]	[ ]	[ ]	[ ]	_____________________
l	l	[ ]	[ ]	[ ]	[ ]	_____________________
m	m	[ ]	[ ]	[ ]	[ ]	_____________________
n	n	[ ]	[ ]	[ ]	[ ]	_____________________
o	o	[ ]	[ ]	[ ]	[ ]	_____________________
p	p	[ ]	[ ]	[ ]	[ ]	_____________________
q	q	[ ]	[ ]	[ ]	[ ]	_____________________
r	r	[ ]	[ ]	[ ]	[ ]	_____________________
s	s	[ ]	[ ]	[ ]	[ ]	_____________________

People who have left Athena:

1	1	[ ]	[ ]	[ ]	[ ]	_____________________
2	2	[ ]	[ ]	[ ]	[ ]	_____________________
3	3	[ ]	[ ]	[ ]	[ ]	_____________________
4	4	[ ]	[ ]	[ ]	[ ]	_____________________
5	5	[ ]	[ ]	[ ]	[ ]	_____________________

Appendix D
Athena Customization Network Questionnaire

Note: Names have been removed for reasons of confidentiality. The actual questionnaire has two pages and includes names from the current staff list, as well as names of people who have left Project Athena.

Name:____ File:________________

Username:_________________________ Date:_______________ Page______of_______

2. For each customization file you use, note the filename above and fill out a separate questionnaire. Identify the sources (people or standard files) from which you received some or all of the file. Also, identify the people to whom you have given some or all of the file. Add comments if you need to explain.

Name	Username	Got it from	Gave it to	Comment?
Public workstation version		[ ]	[ ]	________________________
OLC		[ ]	[ ]	________________________
a	a	[ ]	[ ]	________________________
b	b	[ ]	[ ]	________________________
c	c	[ ]	[ ]	________________________
d	d	[ ]	[ ]	________________________
e	e	[ ]	[ ]	________________________
f	f	[ ]	[ ]	________________________
g	g	[ ]	[ ]	________________________
h	h	[ ]	[ ]	________________________
i	i	[ ]	[ ]	________________________
j	j	[ ]	[ ]	________________________
k	k	[ ]	[ ]	________________________
l	l	[ ]	[ ]	________________________
m	m	[ ]	[ ]	________________________
n	n	[ ]	[ ]	________________________
o	o	[ ]	[ ]	________________________
p	p	[ ]	[ ]	________________________
q	q	[ ]	[ ]	________________________
r	r	[ ]	[ ]	________________________
s	s	[ ]	[ ]	________________________

People who have left Project Athena:

1	1	[ ]	[ ]	________________________
2	2	[ ]	[ ]	________________________
3	3	[ ]	[ ]	________________________
4	4	[ ]	[ ]	________________________
5	5	[ ]	[ ]	________________________

Appendix E
Athena Open-Ended Customization Questions

The following questions are guidelines for the interviewer and are not shown to the study participant. They include three general categories: proactive customizations by the user, situations in which the user has modified their behavior as a result of the technology, and interactions between the two.

1. Part I: What kinds of changes have you made?
   a. If you have been at Athena for a while, you've had the opportunity to try a number of different window managers. How many have you tried? _____________
   b. Do you still use the first one you tried? [ ] Yes [ ] No If not, have you changed your current window manager to be like a previous window manager? [ ] Yes [ ] No
   c. When did you last change your window manager? __________________ Did you "finish"; ie get it working the way you really wanted it to work? [ ] Yes [ ] No
   d. Which of the following changes have you made: [ ] Key bindings [ ] Appearance [ ] Other_______________________
   e. How did you choose the key bindings for your window manager? [ ] Copied from a friend [ ] Programmed them [ ] Use standard
   f. When you log in, do you set up a standard screen layout? [ ] Yes [ ] No
   g. Have you ever changed that layout? [ ] Yes [ ] No If yes, what caused you to change it? [ ] New window manager [ ] Didn't like it [ ] Saw someone else's I liked better [ ] Had a better idea [ ] Other____
   h. What else do you set up when you log in?
   i. Think back to the last few times you have spent an hour or more editing your .dot files. What were the circumstances that caused you to edit them? Please list as many examples as you can.
   j. In general, would you describe yourself as someone who customizes your workstation software a lot or a little? Explain.
   k. Do you customize your workstation as much as you would like? If not, why not?
   l. Do other people around you encourage you to customize your workstation or do they discourage it? In what ways?
   m. At any given point in time, you usually have a choice between modifying your work environment to make it more convenient and using your work environment to accomplish work. What are the circumstances under which you choose to modify? What are the circumstances under which you choose to work?
   n. What would make it easier to customize?
   o. What makes it hard to customize?
   p. Have you used any software for something for which it wasn't intended? Please give an example.
2. Part II: Have you changed your work because of changes at Athena?

a. When Athena converted to 6.3 in September, a number of new "dot" files were introduced, such as .cshrc.mine and .path. Did you edit your "dot" files at that time?

b. Did you create any new ones?

c. Did the use of these new files change the way you organized your working environment, or did you try to make the system operate in the same way as before?

d. Had you known that dot files existed prior to then? _____________

e. Had you customized any dot files prior to then?

f. Do you recall how much time you spent editing dot files at that time? ________________________

g. How often do you spend more than an hour editing dot files?

h. Give examples of the kinds of changes you make.

i. Athena is in the process of converting to 6.4. Have you changed your "dot" files as a result? Which ones and why?

j. Can you think of an example in which learning to use a software application has changed the way you work? (For example, using zephyr.) Please give an example.

k. Can you think of an example in which a change in the way software works caused you to change the way you work? (For example, changing from uwm to mwm.) Please give an example.

l. Sometimes you have no choice about using a new piece of software. Can you think of an example in which the new software significantly caused you to work less well? Are there examples in which it caused you to work better?

m. Has it ever changed what you *want* to do? Give an example.

n. Have you ever looked at how someone else has used a piece of software and suddenly realized that you'd like to do that do?

o. Have you ever looked at how someone else has used a piece of software and thought up something completely different?

3. Part III

a. Have you ever used software in ways that it was not designed for? Please give an example.

b. Did other people use the software after you discovered this new use? Describe.

c. Have you adopted software such changes from other people?

d. How has it changed the way you work?

Appendix F
Consent Forms

Wendy E. Mackay
Massachusetts Institute of Technology
Mail: E40 366
1 Amherst Street
Cambridge, MA 02139

To: The Committee on the Use of Humans as Experimental Subjects:

Re: Application Number 1863 renewal

We have not changed the interview procedure. However, we have added several new questions for the same subject population. In addition to asking participants questions about how they customize handling of electronic mail via Athena Lens rules, we also plan to ask questions about how they customize their other software. Participants will also be asked to list their customization files to show the dates when files were changed and to show the interviewer customization examples of their choice. No new data will be collected automatically. The handling of the data has not changed; it will continue to be held in strict confidence and no information that identifies individuals will be used in any of the reports or shared with the participants colleagues.

We have added Professor Wanda Orlikowski, Sloan School of Management, to the list of individuals who may see the raw data and removed Richard Pito, Brian Gardener, Jeff Siegal, and Toby Mintz, who are no longer working on the project.

Regards,

Wendy E. Mackay

Dear Study Participant:

We are conducting a study of how people customize their application software. The study will help us to understand what people actually do when customizing, in order to design more effective customization mechanisms.

As a voluntary participant in the study, we will interview you about the way you customize your software and ask you to provide us with dated listings of your customization files. We will also ask you to show us examples, of your choice, of your customization files. We will not collect any data automatically.

The data we will collect will be strictly confidential. We will report only summary statistics from groups of users and anonymous examples. Your name or other identifying information will not be included in any of our reports unless you give us explicit permission to do so. we will not share individual information with your supervisor or co-workers. However, the following individuals may see the raw data.

Thomas W. Malone, Professor, Sloan School of Management
Wanda Orlikowski, Professor, Sloan School of Management
Wendy Mackay, Graduate Student, Sloan School of Management, Digital

Your cooperation is appreciated, but your participation in this study is purely voluntary. You may choose not to answer any questions, to stop participating at any time, or not to participate at all without prejudice or penalty. If you have any questions about this study, please feel free to ask us at any time during or after the study.

Because your participation is voluntary, we are required by MIT to document, in writing, your consent to participate in this study. Signing this letter does not waive any legal rights you may have.

Name:____

Signature:_______________________________________

Date:____

Author's Publication List

Refereed Publications

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

Mackay, W.E.,Malone, T.W., Crowston, K. Rao, R., Rosenblitt, D., and Card, S. (May 1989). How do experienced information lens users use rules? *CHI '89 Conference on Human-Computer Interaction*. Austin, Texas: ACM/SIGCHI.

Mackay, W.E. and Davenport, G. (July 1989). Virtual Video Editing in Interactive Multi-Media Applications. *Communications of the ACM*, Vol. *32*(7).

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

Books and Book Chapters

Mackay, W.E. (1981). *Computer-Based Instruction Authoring Guide*. Bedford, MA: Digital Equipment Corporation.

Mackay, W.E. (1986). Interactive Videodiscs: Database Driven Courseware. In M. M. Geerling (Ed.), *Computer Based Education in Banking and Finance*. Amsterdam: Elsevier Science Publishers B.V., North Holland.

Mackay, W.E. (1987). Integrated Learning Environments. In Moonen, J. and Plomp, T. (Ed.), *EURIT 86: Developments in Educational Software and Courseware*. Oxford: Pergamon Press.

Mackay, W.E. (1987). Managing CBI Projects. In Moonen, J. and Plomp, T. (Ed.), *EURIT 86: Developments in Educational Software and Courseware*. Oxford: Pergamon 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.

Other Publications

Mackay, W.E. (1977). *Behavioral Contrast in Pigeons*. Honors Bachelor's thesis, University of California, San Diego.

Mackay, W.E. (1979). *Stimulus Class Formation with an Oddity Procedure*. Master's thesis, Northeastern University.

Mackay, W.E. (1985). *Unpacking Information* (Technical Report 2). Digital Equipment Corporation, Educational Services.

Mackay, W.E. (1985). *Does Tutoring Really Have to Be Intelligent?* (Technical Report 2). Digital Equipment Corporation, Educational Services.

Mackay, W.E. (1987). *Strange Bedfellows: Research Alliances Among Competitive Firms* (Technical Report). Digital Equipment Corporation, External Research.

Mackay, W.E. (1987). Integrated Learning Environments. *Optical Insights*, *1*(1), 5-7. Boston Computer Society.

Ackerman, M. and Mackay, W.E. (May 1989). Context Issues for Users in Multi-Windowing, Multi-Tasking Environments: The Case of X. *CHI '89 Workshop on Context: in the eyes of users and in computer systems*. Austin, Texas: ACM/SIGCHI, Position Paper.

Mackay, W.E. (October 1989). EVA: An Experimental Video Annotator for Symbolic Analysis of Video Data. *SIGCHI Bulletin*, Vol. *21*(2). Special Issue: Video as a Research and Design Tool.

Mackay, W.E. (1989). Video as a Research and Design Tool. *SIGCHI Bulletin*, Vol. *21*(2). Guest Editor of Special Issue.

Mackay, W.E., Gardner, B.R., Mintz, T.H., Pito, R.A., and Siegal, J.B. (August 1989). Argus Design Specification. Distributed by the Open Software Foundation and MIT Office of Technology Licensing.

Mackay, W.E. (September 1989). Pygmalion: A Multi-Media Communication System. Submitted to SIGGRAPH '89 Committee.

Mackay, W.E. (August 1989). Argus Reference Manual and User's Guide. Distributed by the Open Software Foundation and MIT Office of Technology Licensing.

Invited Talks and Panels

Mackay, W.E. (1980). Creating Graphic Displays on Non-Graphic Terminals. *ADCIS'89 Conference Proceedings*. Association for the Development of Computer-Based Instruction.

Mackay, W.E. (1986). Beyond the Wizard of Oz. *CHI '86 Conference on Human-Computer Interaction*. Boston, MA: ACM/SIGCHI, Demonstration presentation.

Mackay, W.E. (1986). Project Athena: A University-Based Case Study. *Laseractive '86*. Laseractive '86, Invited talk.

Mackay, W.E. (1987). User Interface Issues with X. *Xhibition '87*. , Panel presentation.

Mackay, W.E. (1988). Video: Data for Studying Human-Computer Interaction. *CHI '88 Conference on Human-Computer Interaction*. Washington, D.C.: ACM/SIGCHI, Panel presentation.

Mackay, W.E. and Ackerman, M. (1988). *User Interfaces to Educational Software*. Washington, D.C.: ACM/SIGCHI. CHI '88 Tutorial Notes.

Mackay, W.E. (1988). *A Designer's View of the X Window System*. Atlanta, Georgia: ACM/SIGGRAPH. Tutorial Notes Published at SIGGRAPH '88 Conference.

Mackay, W.E. (October 1988). Telecommunications in the 1990's: Human Factors Issues for the Information Age. *Proceedings of the Human Factors Society, 32nd Annual Meeting*. Anaheim, CA: Human Factors Society, Panel presentation.

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., Treese, W., Applebaum, D., Gardner, B., and Michon, B. (July 1989). Pygmalion:

Multi-Media Communication. *Computer Graphics: SIGGRAPH '89 Conference Proceedings*. Boston, MA: ACM/SIGGRAPH, Special event presentation.

Mackay, W.E. and Ackerman, M. (September 1989). Dueling Toolkits: User Interface Issues in Multi-Tasking Window Environments. *HCI International '89: 3rd International Conference on Human-Computer Interaction*. Boston, MA.

Bennet, J., Mackay, W., Conklin, P., Sancha, T., and Guevara, K. (August 1990). HCI Seen From A Developer's Perspective. *Interact '90 Conference Proceedings: Third Annual IFIP Conference on Human-Computer Interaction*. Cambridge, England: Interact '90, Accepted for publication.

Video Publications

Arnold, T., Parkes, P., Mackay, W., and Michon, B. (1983). R&D Animation Research. Videodisc. Experimental videodisc produced by Educational Services Research and Development.

Hodges, M., Arnold, T., Spencer, J., Kulhlman, B., and Mackay, W.E. (1984). Navigation Research Videodisc. Videodisc. Experimental videodisc produced by Educational Services Research and Development.

Mackay, W.E. (1986). Beyond the Wizard of Oz. Videotape presented at CHI'86.

Mackay, W.E. (1988). EVA: Experimental Video Annotator. Videotape Presented at CHI '88.

Mackay, W.E. (1989). Video Editing in Multi-Media Interactive Applications. Video published with July 1989 issue of CACM.