Users and Customizable Software: A Co-Adaptive Phenomenon

WENDY E. MACKAY

Thesis Proposal
Draft: 31 March 90 22:40

Copyright © Wendy E. Mackay, 1990. All rights reserved.

Abstract

I propose to study the use of software that is explicitly designed to be customized. User-customizable software has an important characteristic: it serves to encode individual patterns of use and continues to influence user behavior over time. The possible customizations are constrained by the software design but may also be modified in unanticipated ways. If customization patterns are recorded as files that can be shared among users, these customizations may act to informally establish and perpetuate group norms of behavior.

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. I propose to extend 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.

I will present the results of a pilot study of the Information Lens, a software application that

allows users to customize the process of managing their electronic mail. I will illustrate the theoretical model by describing the development of the Information Lens and identifying the interactions between the technology and individual users in the context of the organization. I will then examine the individual patterns of use of Lens rules and trace the patterns of sharing of rules among members of the organization.

I then propose a study of Unix users at MIT's Project Athena, to examine user customization of software in greater detail. I will interview members of the Athena staff and examine their customization files. I will also examine the questions about customization in the On-Line Consulting System database, which contains over 30,000 questions. I will attempt to identify the factors that influence users to customize as well as the barriers that prevent them from customizing at an optimal level, and I will analyze how users weigh the costs and benefits of customizing their software. I will then trace how users influence each other through the exchange of customization files and show how this affects the organization-wide use of the technology. I will conclude with a discussion of the implications for organizational theory and innovation, suggest changes in the software design process, and propose directions for future research.

Table of Contents

Table of Contents 4
List of Figures 6
List of Tables 7

1. Introduction 8
1.1 What is Customization? 9
1.1.1 The Design of Customizable Software 11
1.1.2 The Perceived Value of Customization 13
1.2 Why Study Customization? 15
1.3 Research Strategy for Studying Customization 19
1.4 Glossary of Terms 21

2. Theoretical Framework 23
2.1 Competing Research Strategies 23
2.2 Structurational Model 26
2.3 A Structurational Analysis of Co-Adaptive Phenomena 28
2.4 Research Strategy 33

3. Illustration of the Structurational Model: The Information Lens 36
3.1 What is the Information Lens? 36
3.2 Research Study 38
3.2.1 Research Site 40
3.2.2 Participants 41
3.2.3 Data Collection 43
3.3 Lens Use in the Context of the Structurational Model 47
3.3.1 What is Lens, really? 47
3.3.2 Arrow ***a***: Users appropriate Lens 48
3.3.3 Arrow ***b***: Users are affected by Lens 53
3.3.4 Arrow ***c***: Institutional Properties Influence Lens Users 57
3.3.5 Arrow ***d***: Lens affects institutional properties 62
3.3.6 Arrow ***e***: Lens use affects group norms of behavior 63
3.3.7 Arrow ***f***: External events affect user's choices 77
3.3.8 Arrow ***g***: Individual factors affect user's choices 85
3.3.9 Arrow ***h***: User's choices affect individual factors 86
3.3.10 Arrows ***i*** and ***j***: Influences on future Lens development 89
3.4 Conclusions 95
3.4.1 Implications For Software Design 97
3.4.2 Implications For Organizational Theory 99

4. Research Method 102
4.1 Proposed Research Method 102
4.2 Research Site: MIT's Project Athena 102
4.2.1 Organizational Structure 104

4.2.2 Hardware environment 105
4.2.3 Software environment 107
4.2.3.1 On-Line Consulting System 111
4.2.3.2 Choices about Software Technology 112
4.3 Study One: In-depth Examination of Athena staff 114
4.3.1 Participants 114
4.3.2 Procedure 116
4.3.3 Proposed Analyses 117
4.4 Study Two: Longitudinal Study of MIT Community 121
4.4.1 Participants 122
4.4.2 Procedure 123
4.4.3 Proposed Analyses 123

References 124

Appendix A. Proposed Thesis Outline 128

Appendix B. Information Lens Questions 132

Appendix C. Ratings of Information Lens Features 134

Appendix D. Athena Customization Questionnaire 135

Appendix E. Athena Communication Network Questionnaire 137

Appendix F. Athena Customization Network Questionnaire 138

Appendix G. Athena Open-Ended Customization Questions 139

Appendix H. Sample OLC Log files 141

Appendix I. Consent Forms 146

List of Figures

Figure 1-1:	Net Benefits of Customization versus Time Spent Customizing	1.1.2
Figure 2-1:	Structurational Model of Technology (Orlikowski 1989)	2.2
Figure 2-2:	User-centered view of the Structurational Model: Inputs	2.3
Figure 2-3:	User-centered view of the Structurational Model: Effects	2.3
Figure 2-4:	Relationship to Structurational Model	2.3
Figure 3-1:	Structurational Model: Lens Example	3.3.1
Figure 3-2:	Version 3: Folder Tree	3.3.3
Figure 3-3:	Conversion from version 2 to 3: Users A and B	3.3.7
Figure 3-4:	Conversion from version 2 to 3: Users C and D	3.3.7
Figure 3-5:	Conversion from version 2 to 3: Users E and F	3.3.7
Figure 3-6:	Conversion from version 2 to 3: Users G and H	3.3.7
Figure 3-7:	Conversion from version 2 to 3: Users I and J	3.3.7
Figure 3-8:	Original Information Lens Rule Editor: Version 1	3.3.10
Figure 3-9:	Revised Information Lens Rule Editor: Version 3	3.3.10
Figure 4-1:	Project Athena: Organization through 1989	4.2.1
Figure 4-2:	Project Athena: Reorganization in January 1990	4.2.1
Figure 4-3:	Factors that affect the user interface	4.2.3
Figure 4-4:	Exchange of customization files at Project Athena	4.3.3

List of Tables

Table 3-I: Dates of Interviews at Research Site 3.2.3
Table 3-II: Patterns of Rule Changes From Version 2 to 3 3.3.7
Table 4-I: Customization Options 4.2.3
Table 4-II: Preliminary Results from Customization Questionnaire 4.3.3

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.

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.

1.1 What is Customization?

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 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 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

---

[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?'"

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, rather like tracing the spread of genetic traits throughout a population. (Contrast this with other forms of communication, which may affect people at particular points in time but do not exert a continuing influence.) 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.

1.1.1 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.

1.1.2 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.

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. An unsuccessful attempt takes time and may introduce problems, resulting in a 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

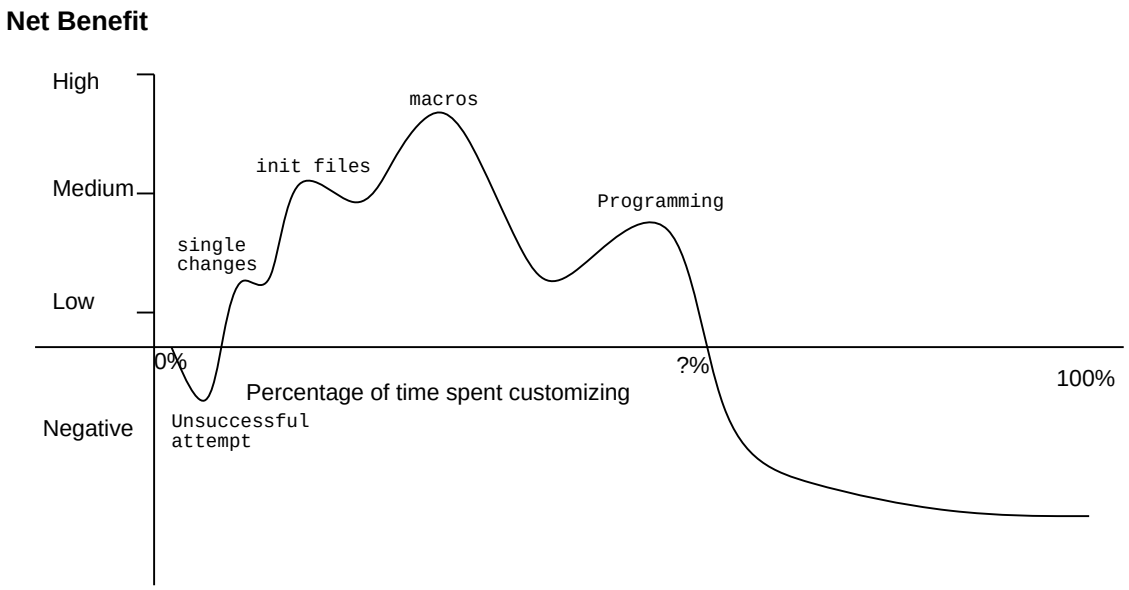


Figure 1-1: Net Benefits of Customization versus Time Spent Customizing

satisfaction), and will begin to interfere with work. Learning to program (for the purposes of customization as opposed to accomplishing a job) has 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. Unfortunately, users do not know in advance how long each type of customization will take, nor do they fully appreciate the benefits. 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 the costs and benefits against a threshold, which may change over time. 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.

1.2 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. The factors that trigger or prevent users from deciding to customize 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. Learning how to affect the triggers and barriers for customization should be applicable to these other areas as well.

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 organizations encourage productive user interaction with their technologies.

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

[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.

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.

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 or 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. One of the questions I would like to explore in this dissertation is whether customizations that affect the user's *process* of working are more likely to result in real innovation than those that simply affect functional characteristics. Preliminary evidence from the Information Lens study and the Athena studies indicate that users are more likely to view a change as an innovation if the change results in a new set of work behaviors.

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.

This is an example of a principal defended by Abelson and Sussman (1985) who 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.

1.3 Research Strategy for Studying Customization

I propose to examine how users customize software and the effect this has on future adoption of new technologies. This work will have an impact on our understanding of how users incorporate software tools into their work lives as well as suggest 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, I propose to frame my research within the model proposed by Orlikowski and extend her analysis of how individuals interact with technology within their organizational settings.

I propose 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 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 the proposed work.

1.4 Glossary of Terms

Normal use — The use of application software as specified by the software design, with default values specified for customizable elements.(p.10)

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.9)

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.10)
Customizable Software	Software that can be customized by an end user through mechanisms provided by the programmer. Customizations affect user behavior over time.(p.9)
Flexible Software	Software that allows users a wide range of interaction choices within the software's basic design.(p.12)
User Innovation	Novel or unanticipated customizations, software modifications or uses of software that provide a positive benefit.(p.17)
User	Any person who interacts with a software application, including end users and programmers.(p.10)
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. 10)
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.10)
Application Software	Software designed to help end users accomplish various tasks, including text editors, spreadsheets, and CASE tools. (p.11)
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.16)
Multi-tasking	The ability to interact with two or more software applications simultaneously on the same computer (usually a workstation).(p.16)

Chapter 2

Theoretical Framework

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

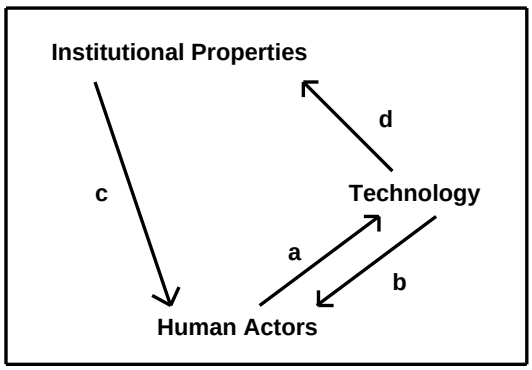
recipients of information. They *react* to information, which changes their behavior and thus the behavior of the system. More importantly, they are sometimes *proactive*: they look for information from the system. They bring assumptions and experience and apply them to the technology in both anticipated 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)

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.



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)

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.
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 Co-Adaptive Phenomena

The research proposed here fits within the theoretical framework advanced by Giddens and Orlikowski, with special attention paid to the interaction between human actors (users) and the technology (application software). I will examine both how users are influenced by the existing technology and also how they affect the technology, through customization or user innovation. I will also explore how users in a group affect each other's patterns of behavior through the exchange of customization files, resulting in the establishment and perpetuation of standards of behavior.

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.

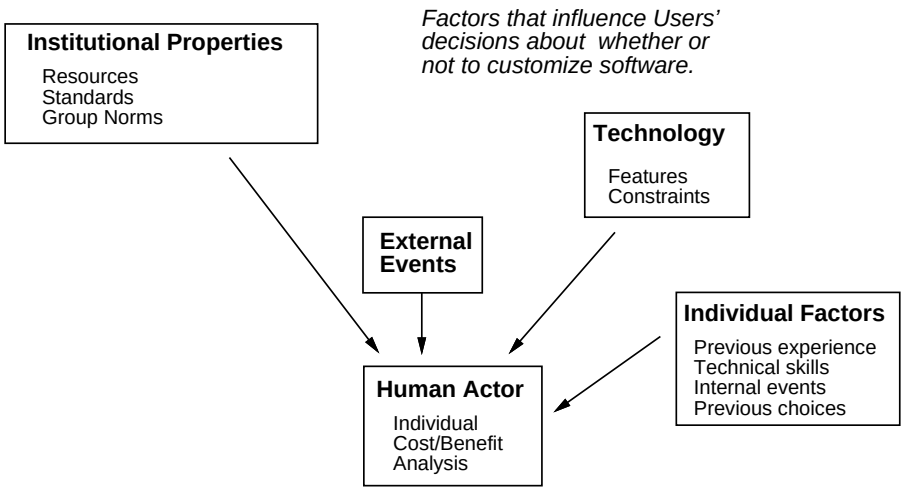


Figure 2-2: User-centered view of the Structurational Model: Inputs

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 2-4 relates this user-centered perspective to Orlikowski's model; heavy lines correspond to the arrows in Figure 2-1. I will examine the influences represented by arrows a, b, and c, but will not explore the effect of the technology on the institutional properties, arrow d, except through the actions of individuals. In addition, I will explore the

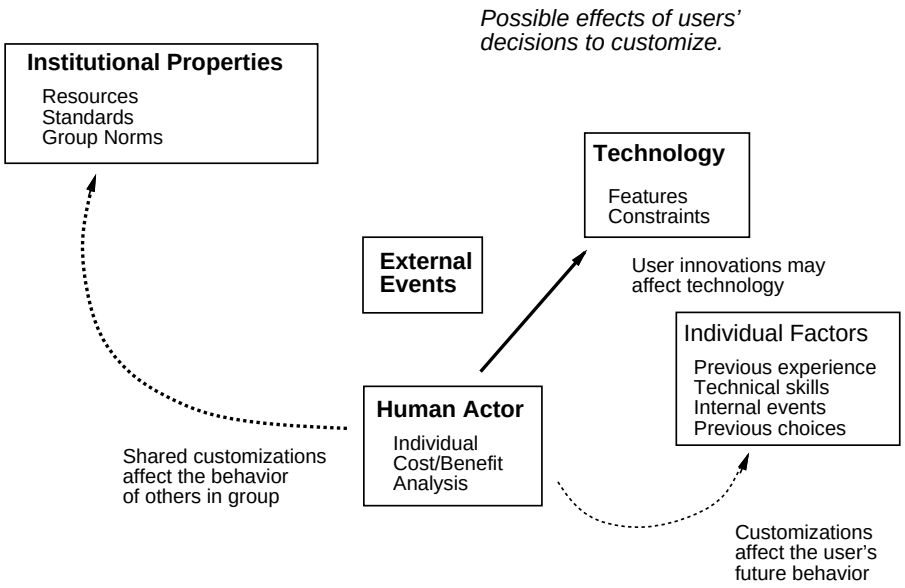


Figure 2-3: User-centered view of the Structurational Model: Effects

influence of other external factors that were not included in the structurational model. I will concentrate on examining the two sets of dotted line pairs. The first set represents the effects of the user's decisions on future behavior. The second set represents the effects of sharing customization files among members of an organization.

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

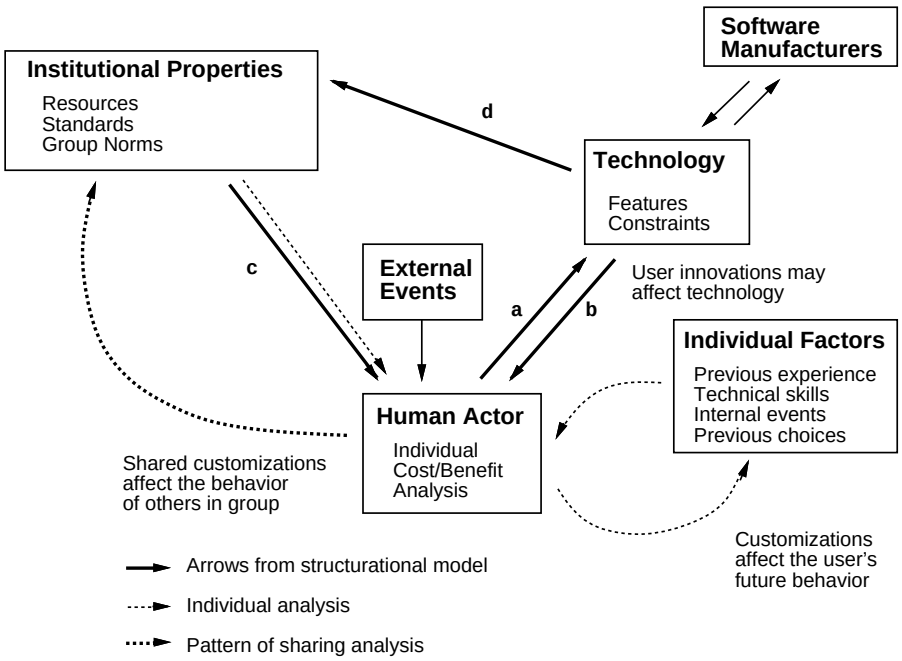


Figure 2-4: Relationship to Structurational Model

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 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."[3] (NCAR 1985 Annual Report, p.14, p.65)

---

[3]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 which they 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.

2.4 Research Strategy

I am 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. I will also examine how users customizations affect the user's 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. Some of the proposed research methods for the thesis were used in this study and will be described accordingly. Chapter 4 describes the proposed research methods and analyses for the thesis in greater detail.

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 the 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 1987-1988 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 in the

system 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 arrived shortly after this time and interviewed these users about their experiences with the software.

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 accomodate 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 less commonly used. 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 identitites 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[4], 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.

3.2.3 Data Collection

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

[4]The secretary used Lens for both her personal mail and her manager's mail.

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. The actual questions are listed in Appendix B.

Interviews were conducted over a number of visits, shown in table 3-I. The last three visits in phase two included a combination of preliminary, installation and follow-up interviews.

Table 3-I: Dates of Interviews at Research Site

Phase	Start		End		Year	
1	10/13	(Mon)	10/14	(Tue)	1986	
1	1/27	(Tue)	1/29	(Thu)	1987	
2	7/20	(Mon)	7/24	(Fri)	1987	
2	8/20	(Wed)	8/26	(Wed)	1987	Kickoff
2	9/1	(Tue)	9/4	(Fri)	1987	
2	11/5	(Thu)	11/6	(Fri)	1987	
3	4/14	(Thu)	4/19	(Tue)	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; the rating list is shown in Appendix C. 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 used 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 missunderstand the process by which the software itself is changed through design and use. Lens is 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. 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.

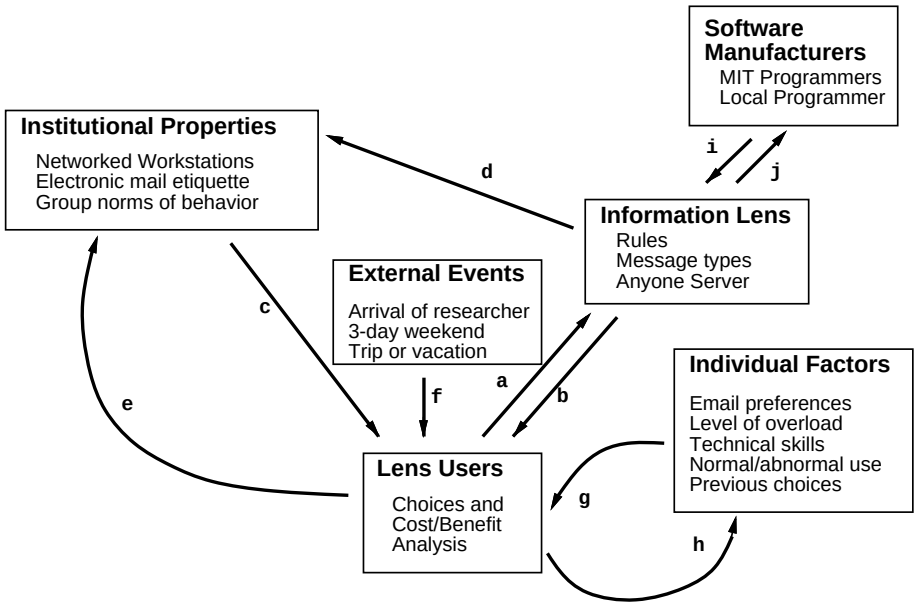


Figure 3-1: Structurational Model: Lens Example

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 this group used the existing electronic mail system in diverse ways. In addition to the normal functions of a message system, they used 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 a manager, who wanted to use Lens rules to help him communicate with his secretary. He objected to using formally-defined message types, because they were slow and none of the existing types were appropriate for his purposes. So, he used a customization feature in a slightly different way to accomplish the same objective. Users were not permitted to create their own message types, however they were allowed to create a list of alternatives to insert into a field when composing a message of a particular type. For example, a user might want to specify a list of people that he or she commonly send messages to; these would then appear as options in a menu that could be inserted while composing a message. The manager felt that he could categorize his messages into five types, which his secretary could then act on. Rather than creating five different message types, he added five code words to the menu for a particular field: *Please file*, *Remind me*, *Action*, *Please reply to*, and *fyi*. He could then read his mail, identify a message that he wanted to forward to his secretary for action, quickly append one of these code words to a field in the forwarded message, and send it to her. She could then use her Lens rules to identify these code words and act accordingly. This is interesting both as an example of how the system can be used to support communication between two people who work together closely, but it also demonstrates the user's desire to annotate incoming messages. She could not have created rules that identified the actions he appended to the message without his intervention -- the actions were independent of the content of the message.

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 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. Other users heard about this feature and several joined the study and began to use Lens in this way. Because of the success of this feature, version three was designed to explicitly support multiple rulesets. Thus, this became an example of a user-initiated change that affected the behavior of others in the organization (3.3.6) as well as the future design of Lens (3.3.10.

Version three provided users with two types of rulesets: "new mail" rulesets were triggered automatically when new mail arrived and "folder" rulesets were triggered manually by the user. In addition, users could create and name additional rulesets and run them manually as well. Previously, it had appeared that most users belonged to one of two camps: those who wanted Lens to "prioritize" their mail and those who wanted Lens to sort and "archive" previously-read mail. However, once users had the option of using as many rulesets as they wished, many people used rules in both ways. Some created two, three or more rulesets to be run at different times for different purposes. 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. fire upon receipt of new mail) were clearly useful, manual triggers (i.e. 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 will simply run the ruleset.

This observation is an example of a more general problem: users are not very good at articulating their behavior in sufficiently explicit terms for the computer. A contributing factor is that people rarely have access to data about their own behavior. Thus their perceptions of their own behavior are often wrong. 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. 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). Allowing users flexibility in specifying when to run rulesets allows them to successfully take their current work context into account without forcing them to be explicit.

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 of 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. 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 (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.

What is Lens, really? The definition has changed again. Users appropriated the technology, which influenced a subsequent version of Lens. Users appropriated the new version, providing new insights into user's behavior and the usefulness of the software.

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. These users 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.". 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: "It would be very interesting if it could send me about ten messages a day." Others didn't: "It will only encourage more junk mail."

As mentioned in section 3.3.3, 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: "Sorting seemed most useful at first, then you poke around

and discover things." The conversion between version two and version affected users perceptions directly. In version two, everyone had to choose between a sorting or "archiving" strategy and a "prioritizing" strategy. When these users recreated their rules for version three, most people considered whether or not to use more than one ruleset. Some people who had been adamant about one strategy, decided to try both types of rules. A few created additional rulesets for special purposes, such as returning from a trip. One person, user ***H***, switched completely from one style to the other.

User ***H*** had been reasonably satisfied with Lens in phase two. His rules were designed to sort mail after he had read it. 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 single new rule that identified messages addressed directly to him. When he returned, he found that this rule 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 rules. He subsequently added one more rule, which identified messages about the conference and forwarded them to his assistent. He said these two rules "changed my life". He thus converted from a post-reading sorting strategy to a new-mail prioritizing strategy. Although he intended to recreate an additional ruleset for sorting rules, he never got around to it. 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. He could put off viewing the other messages until he had time. The net result was that, by reducing the number of rules and running them at a different time, he was able to significantly alter his approach to handling mail.

People reported a variety of changes in their behavior, both of use of Lens and in their perceptions of what they do:

- "I am pickier about what I read now and have attended more forums. Without Lens, I probably would have deleted those messages."
- "I read mail less often than I did before and I now go to [lab] meetings that I would have missed before."
- "I used to read, then sort. Now, I sort directly."
- "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]"
- "I dropped off several d.l.'s. [Lens] made me realize I was getting mail from useless d.l.'s."
- "I use the customized pop-up alternatives less than I expected -- the buckets are too big."
- "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'."

Lens is more than just the advertised features of rules and message types. It includes a number of features of the implementation 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, developed for version three (Figure 3-2. Users could easily create and move folders in the hierarchy. Of the 23 people who tried it, all but three reorganized their mail folders from a single level to a tree structure with two to four levels. Some people used this conversion as an opportunity 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. However, a few people tried it and 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."

---

[5]Distribution List

Insert picture of Phase Three Folder Tree here

Figure 3-2: Version 3: Folder Tree

Sometimes the effects of using Lens change over time, as the user's job situation changes. I interviewed a programmer who had recently joined the organization (although not the company). At the preliminary interview, the person was overwhelmed with email, had subscribed to 46 distribution lists and wanted Lens to help him deal with the problem. Six months later, the person had stopped using Lens. He explained that he had figured out which distribution lists were useful and "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 back in control.

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 was important to identify the behavior patterns of individuals in order to discover that the technology had been appropriated in these ways. Individual

analyses are particularly important when the behavior is highly variable (making summary statistics highly misleading) or when individual patterns cancel each other out.

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 (and found it 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.

We can't evaluate rule strategies unless we look at individuals. From there, we begin to see different patterns of behavior and different reasons for different activities. Once we pull out these reasons, we can look more globally to see if these patterns hold.

The purpose of examining individuals is to understand how people make unique decisions about customization -- and what affects those decisions. The proposed theoretical framework allows us to look for both how the user is influenced by new technology and how the user appropriates the technology for his or her own purposes. By studying individuals, we will be able to identify the set of triggers and barriers that affect the use of new technology. By studying the organization, we can see how these individual patterns affect the behavior of the organization as a whole.

3.3.4 Arrow *c*: Institutional Properties Influence Lens Users

1. Message forms, NEXT
2. Viewpoint user interface
3. Mail etiquette affects delete rules
4. Opposition to "flamers" results in idea for boring rule
5. Existing message types

Arrow ***c*** in Figure 3-1 represents the effect of institutional properties, especially group norms of behavior, on individual users.

Institutional properties consist of the characteristics of the work and social environment in the lab. The fact that everyone uses a workstation connected via a high-speed network, that electronic mail is an essential medium for communication, that certain kinds of behavior are acceptible and other kinds are not. Certain kinds of activities are routine: for example, phone messages are left with a receptionist, who sends the recipient an email message. Everyone who handles a visitor is supposed to fill out a visitor's log form. The lab invites outside speakers to talks on a regular basis and everyone in the lab is invited. These factors had various influences on how people used Lens.

After one user discovered the ability to run rules *after* reading mail, described in section 3.3.2, other people quickly heard about it and some who had previously refused to use Lens decided to try it. These people created a set of rules designed explicitly for sorting their mail. Version three explicitly supported multiple rulesets. When this group converted to version three, many of them created both "new mail" rulesets and "folder" or sorting rulesets. Thus, they learned of a new use by talking to other Lens users, started using it, and then realized that the style of use they had previously objected to was, in fact, useful.

Asked members of the first group why they tried it -- some said the talk, the ideas, and peer pressure. Version 2 ... talk, perception of more support, use of software by several lab managers. Version 3 -- knew programmer, respected his work, word of mouth.

Several people commented that they did not like the early versions of Lens because their friends reacted poorly to the extra fields, especially if they were empty: "[People] didn't like getting all these strange headers" and "the header has garbage that nobody wants to see".

They didn't like the fact that they had no control over how the final message looked. They reacted better to version three because "it looks more like people generated [the message], not a machine."

Another comment from early users was that the user interface to Lens was not close enough to what they were used too. One commented: "The differences from Lafite are annoying." and a number of others requested that a particular key, *Next*, operate as it did in that editor. Part of version two was developed at the research site and users praised it as being much better on these grounds than version one. Version three was written by a member of the organization and fully supported the Viewpoint user interface standards. (A few programmers objected to this interface and preferred using LISP instead.)

People also wanted to use the message types that already existed in the organization. Users received relatively standard phone messages from the receptionist (one programmer had written his own LISP rule in Lens to extract these messages and pop them up on his screen). Other commonly used message types included bug reports (which saved the sender time by automatically filling out critical information about the system), seminar announcements and a "visitor log" that everyone was asked to send in to help keep track of the number of visitors.

One of the most striking examples is the lab's unwritten rules of etiquette. Many people spoke of how others violated these rules in their use of electronic mail and a number of people cited these violations as one of the key reasons for using Lens. The most common complaint was the indiscriminant use of "cc:" in which people...

Also, people in this lab are placed on distribution lists without their knowledge. Some of these lists are based on membership in a group or organization, others are based on

geographic location. Most of the mail sent to these lists are "broadcast" messages of which the user is generally interested in only a small subset. When people use these lists for trivial purposes, people get upset. ""

Distribution lists have as many unifying characteristics as the characteristics that unite any group of people. Thus, it is not possible to talk about messages from these lists by themselves -- it must be qualified with the kind of dl. For example, a list composed of four members of an active working group who are exchanging work-related information and delegating tasks several times a day may be treated as more important than personally-addressed messages. In contrast, a list based on the participant's geographic location is less likely to contain information of interest. (Thus, one sees "lost car keys" and "fire alarm test tomorrow" kinds of messages.)

Special interest distribution lists are an interesting subgroup. People usually volunteer to go on these -- like reading a special-interest bulletin board or joining a news group. People also often forget about which ones they are one -- especially if traffic is low. These lists develop 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. People evaluate how much of this they are willing to tolerate. 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.)

One person commented that you have to watch carefully, because "good lists go bad.")

Some people objected to the Lens *anyone* server, because it would encourage people to send out additional messages of this sort. "There's too much junk already."

Members of this lab have a strong sense of etiquette -- how people should interact with each other. This appears both in terms of accepted use of electronic mail and in use of the network resources. Many commented unfavorably about breaches in etiquette and talked about Lens as a method by which they avoid the consequences of rude behavior.

Work environment affects how people work. Issues of email etiquette, position in the organizational hierarchy, etc.

Several people commented that they didn't like sending Lens messages, because other people thought that the extra, unused headers looked funny. Another commented about message types -- "too simplistic", and "request for action" is silly if the message is asking to borrow a book.

Autogetmail and Network Etiquette

People commented that they really liked autogetmail. One person even used it as the main feature of Lens and never ran rules. People commented about its effect on their behavior: "My mail reading pattern changed -- now I read it when I want to, not when it arrives."

But, in a later interview (4 months later), several people had stopped using it: "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." These were technical people who realized the effect of their actions on others in the organization. However, less technical people, who didn't realize this, continued to use it. "Autogetmail is really greal...I would be offended if I had to get my own mail."

3.3.5 Arrow *d*: Lens affects institutional properties

1. Only through use
2. Creation of new procedures to increase effectiveness of Lens
3. Note: institutional properties affected the design of Lens: no bulletin board, limited msg types, LISP-based rule system.

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 the technology in ways that affect the organization. Because of the size of the study, it was difficult to determine any effects of Lens on the organization. Features that might have been expected to have an effect, such as message types and the anyone server, were never used on a regular basis.

Some people were concerned about the potential impact of Lens features on the organization: "Most people are too arrogant and most mail is junk... *anyone* would generate more." There is evidence that if the anyone server introduced a significant load on the organization, it would be stopped. 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.

Although not shown in the diagram, some of the properties of this organization and its use of computing resources affected the intitial design of Lens. Because the software environment did not support bulletin boards and used distribution lists for everything, the concept of the anyone server was born. Although a rule sorter had already been created at this site (Levy, 1986), the interface was difficult and only programmers reported it useful. Thus, the idea of creating an editor for rules that would look like a message and be easier

to use was born. Finally, the mail system already supported personal message types, and certain kinds of message types were already used for reporting bugs and keeping track of visitors. So extending the concept of message types and allowing the organization to explicitly create them in order to increase the effectiveness of rules made sense.

A characteristic of electronic mail is that people receive more messages than they send. Thus, it makes sends to provide senders with incentives to provide more useful messages since, from the perspective of the organization, it is more efficient. Unfortunately, senders have few personal incentives to take the extra time to use an appropriate message type and save the recipient time.

Certain kinds of messages are very useful for the recipients -- others are more useful for the senders. Thus, the receptionist who sends out phone mail messages all day is more likely to appreciate and use a phone message type than a staff member who send such messages rarely. Other kinds of messages are also sent from a single source, e.g. one person tends to send out seminar announcements. It doesn't make much sense to give everyone in the lab the ability to send these announcements if they never do (because the more choices of message type, the less likely people are to use them). People objected to the number of message types available -- they simply wouldn't choose to send most of them, even though they would like to create rules for when they receive them. Thus the organizational structure affects what makes sense in terms of Lens rules.

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

1. Patterns of sharing rules
2. Retaliation against people who abuse mail (boring, lif, ks)

Arrow ***e*** in Figure 3-1 represents the effect of of individual users on group norms of

behavior and other institutional properties. The key finding here is that people share knowledge and ideas about how to use the technology, and some people even share the rules themselves.

From the very beginning, people asked for examples of rules. "I want a Lens wizard...I'm busy and have a low tolerance for learning new things. I'd like a profile written by a wizard." "I didn't write any rules...[I] couldn't think of anything useful."

Although I wasn't looking for it explicitly, it became clear that people wanted to share their rules and that, even though the software did not support this, they did. I have incomplete records, but have gone back and spoken to some of the people, also have records from interview data and have traced the paths of several rules that were created by one person and shared with others.

In two instances, one person simply copied all of someone else's rules. In one case, the rules proved not useful and the person stopped using Lens. He later used those rules for sorting mail after returning from vacation and found the rules very useful. This experience caused him to start using version three of Lens. When he converted, he created a completely new set of rules and then "borrowed" specific rules from the person he'd copied from before. The other person who borrowed rules also edited them, but kept the same basic style.

People developed different rule "styles". Some people concentrated on creating a small number of rules for efficiently identifying important messages (either by removing them from the inbox or by moving all unimportant messages out of the inbox). Another style was used after the user returned from a trip. "When I'm back from a trip, I get tough. I'm vicious about messages... I wipe out seminars, etc." Another strategy was to use Lens to sort messages after they had been read.

In studies of the use of Argus, several people developed a two-step strategy, with one set of rules for sifting and another for subsequent filing after the files had been read. The patterns of use of Lens rules at Athena were quite different than at the previous research site, because the people shared the ideas about it. also, the software didn't name rulesets "folder" and "new mail". So people would create a ruleset with one rule, and others with several. Many more rulesets here. This speaks to the evolution of the software as well -- users are influenced by how people talk about it. One effect of my first paper, which identified two strategies (prioritizing and archiving) allowed people to evaluate their own behavior and identify themselves as using one or both strategies. However, this also served to limit their ideas about how to use rulesets in these ways -- few people used them in other ways, such as the two-step strategy I mentioned.

Lens may be seen as a diagnostic tool, to discover the kinds of activities the organization is engaged in and what it considers important. It can be a diagnostic tool for finding out what the organization does. Note, too, that the ability to record data (a tool for researchers), has an important impact on the user's reflection about his or her use of mail -- which causes change in and of itself.

Individual factors:

One thing that is abundantly clear is that people use mail in highly diverse ways. They have appropriated for a variety of purposes, including time management, efficient pointers to and storage of technical information, and task delegation within a group.

Studied lens: found that people coopted that system, the innovations spread to others. Also found that some of the ideas about Lens weren't adequate to understand what people did and why -- needed to observe use to understand (eg context). Also, issue of learning:

the very act of asking people to think about writing rules about email changed their way of thinking about it and using it -- which means they had learned one way (or a limited set of ways) of thinking about email, coopted it a bit, and then, when presented with a new model of use, coopted that. They thus demonstrated that they learned from the system, changed it, then learned again, and changed again.

This relates to the more general problem of making things explicit for the computer. Most people do not "know" the rules they use for managing their electronic mail. 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 the behavior of the system as well as the user. It is thus not surprising that people find it difficult to articulate the rules they want and need for processing their electronic mail -- although they often can come up with examples of tools that would make things easier or solve a particular problem. When designing the system, providing a means for allowing users to direct the system without explicitly articulating rules, is likely to be more useful. Will this result in more innovations, because the users are likely to use the system based on their actual, rather than articulated, use and thus discover new aspects of their behavior and the system?

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-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. This characteristic of Lens had nothing to do with the ideas or even the basic implementation of Lens, but had a significant impact on how it was used.

Managers and high-level types generally delegate via email. A small percentage of their email requires an answer, because they get copied on things for political reasons, etc. Sec can answer many automatically. also get requests for vacation, meetings, etc. If the manager doesn't answer, no problem for manager -- the requester needs something from the manager not the other way around.

However, other people, especially those in support positions, must reply to email. If the person who has to fix broken sw doesn't reply, it means that he or she isn't doing his or her job. Secretary must respond to bosses requests. So the whole social order is reflected in the rules these people write. A manager is more willing to "miss" important messages (false rejection vs miss).

Delete

People use different deletion strategies. They also create more qualified delete rules than other rules. 60% of delete rules were qualified whereas only 32% of move rules were. Only 38/190 = 21% rules were delete rules.). Very few delete a person (5), and those are qualified.

Social norms affect deletion strategies... Some people operate outside the expected norms

of behavior. Within the org, since email is always identified, people learn that it's a bad idea to "flame" and there's quite a bit of social pressure against excessive use of email. Some people outside of this get either don't understand (KS) or actually abuse it (LIF). Who communicates with whom, who sits outside of the organization (Lifschutz, ks). What people have to respond to email.

"I use the boring rule to filter out flamers."

If we evaluate the four possible situations in deleting mail, the results of a particular deletion decision can be evaluated as one of the four:

1. correctly deleted: Correct rejection
2. Incorrectly deleted: Miss
3. Correctly kept: Hit
4. Incorrectly kept: False alarm

People's situation in the organizational hierarchy, and the ways in which they have to respond to email, affect where they place their cut-off. Most managers can afford to incorrectly delete a message (it will be sent again or someone will call or stop by). They are thus more interested in reducing the number of messages that are incorrectly kept. People who conduct much of their work via email or who cannot refuse most of the tasks delegated in this way, generally have a higher level of incorrectly kept messages so as to lower the possibility of incorrectly deleting something that might be important.

User perception is not equal to reality (user's don't really know what their "real behavior is"). Rules are based on perceptions, not reality. Presenting users with data about their own behavior changes it.

Patterns of Sharing Rules

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

The activity of customizing one's own software applications might be viewed as a reasonably solitary task. After all, what an individual does in the privacy of his or her office, for reasons of personal taste and efficiency, appears to have little to do with the rest of the organization. However, because people have varying levels of desire and ability to customize, and have limited amounts of time, they may 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 and reduce 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.

Do individual customizations of software affects user interactions in the workplace? At first, the answer would probably be "no". After all, what an individual does in the privacy of his or her office, for reasons of personal taste and efficiency, appears to have little to do with the rest of the organization. However, one of the key mechanisms for adopting customizations is through copying other people's files. This has an on-going impact on both the communication patterns within the organization and encourages standard patterns of behavior. Both useful innovations and errors can be propagated throughout the organization in this way.

Another social/organizational impact occurs when customization is used to improve the communication among members of a group. In the Lens study, a self-identified group customized the lens rules to make their interactions with each other more efficient. In another example, manager/secretary communication patterns that were set up to match the particular forms of communication that each set preferred. (Describe both examples.)

From a research standpoint, customization files are interesting to study because the users' patterns of use are encoded as artifacts that can be traced throughout an organization, rather like tracing the spread of genetic traits throughout a population. More importantly, the use of these files continues to affect daily patterns of behavior over time. Systematic study of records of changes, correlated across the individuals in an organization, will provide a view of a communication network within the organization that has on-going impact over time.

Purpose: discuss the social aspects of an apparently non-social practice, that of customizing one's software environment. I will trace how customizations spread through an organization and the on-going effects this has on the behavior of the organization. I will conclude with recommendations for the design of customizable software, e.g. that developers should expect that users will share their customizations, for better or for worse, and consider the impact of this on the long-term use of the system.

Exchange of Information Lens Rules

Although some individuals in the organization created their own unique sets of rules, many shared rules with each other. 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.
"Lead user"	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.

Not surprisingly, the person with the most visible effect was the researcher, who gave most of the Lens users a preliminary set of five examples of rules.[6] These rules acted as default rules which users could customize for their own purposes. The rules were chosen to illustrate rules that others in an earlier study had found useful, as well as point out a technical trick that users might find helpful. The sample rules were included in a five-page "mini-manual" which consisted of a one-page summary of the Lens features, a page of the five sample rules, and a three-page "day-in-the-life" example of how someone would use all the features throughout the day to solve a "typical" set of mail filtering problems.)

The programmer who created the local version of the Information Lens, which was more readily accepted by the members of the lab, also shared rules. He 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, but these were incorporated into the design of the software and made available to everyone.

One of the non-technical members of the staff was intrigued by the strategy of one of his colleagues and created his own set of rules based on that strategy. He then provided a complete copy of the rules to another of his (non-technical) colleagues.

The most interesting person was the earliest and most steadfast user of Lens. One of the top technical members of the lab, he immediately began experimenting and created almost

---

[6]People could choose to use Lens without participating in this study.

100 rules. Throughout the project, he conferred with the developers and made numerous suggestions to the developers; he was also vocal with complaints. He created a number of innovations that were incorporated into the local version of the software.

This person created a number of complex, but useful rules that could be traced in other people's rulesets. A good example is the "Boring" rule. He had noticed that some messages (usually "flames") result in a long string of replies. If the original message is not interesting, chances are very good that the replies will not be particularly interesting either. Fortunately, the mail system identifies these replies by placing "RE:" in the subject field and appending the contents of subject of the original message. For example, if the orginal message has the following subject: Tyrolian Ski Bindings, then subsequent messages will have the subject: RE: Tyrolian Ski Bindings. He added a feature that allowed users to identify a potentially boring message and creates a Lens rule to automatically delete it. (For example: If subject: RE: Tyrolian Ski Bindings then delete message.)

Another example was the identification of a well-liked author in a commonly-read jokes distribution list. He created two rules. The first identified and saved jokes by this author; the second identifies all other jokes and deletes them. This combination of rules also proved popular and was shared by x of his colleagues.

Most people were uncomfortable with the idea of automatically deleting messages from a particular sender, even if they were quite sure that the sender would be unlikely to send anything important. Many people in the lab were annoyed by a person outside the lab who had discovered an internal mailing list and was using it to announce various seminars. "Fred" created a rule that effectively deleted all of the seminar announcements without risking the deletion of potentially important messages.

In each case, these rule appeared in other people's rulesets. They were copied and used, and sometimes discarded. Most of the people who copied the rules were programmers with quite long lists of their own rules. These people ususually adopted one or two rules at a time. One of his colleagues adopted his entire set of rules; but found they weren't appropriate. This person stopped using Lens for many months, and then rediscovered its usefulness when filtering mail after returning from a trip.

People adopt some of his rules. you can tell who talks to whom. Points about gatekeepers. About the benefit of having "artifacts" -- unlike gatekeepers, people see things and someone becomes the person responsible for handling it. Not necessarily the developer -- unless the developer is an innovative user as well. Can trace the path of influences among people. Who's tied in, who isn't. Implications??? Facilitate. Problem of copy vs. point to -- do you use someone else's stuff "as is" and let them maintain it? Benefit: they fix bugs, "take care of it". BUT, if their use diverges, or they move, you're stuck.

People usually let someone else in the organization act as the "first user" -- and let that person play. Then this person becomes the local source of ideas -- you send nifty rules to this person and ask for his/hers. Also, when you see other peoples rules that look interesting, chances are that the rule when through this person at one time.

Another social/organizational impact occurs when customization is used to improve the communication among members of a group. In the Lens study, a self-identified group customized the lens rules to make their interactions with each other more efficient. In another example, manager/secretary communication patterns that were set up to match the particular forms of communication that each set preferred. (Describe both examples.)

Customizations that turn into user innovations may well result from situations in which groups attempt to modify the software to meet their needs. In the previous example of manager/secretary relationships, the manager decided to use customization fields in the display of the rule system to create a mechanism for communicating in a standard/automatable way with his secretary.

Although the number of users was quite small, and there were no facilities for exchanging rules, it appeared that users did in fact share rules with each other. It also appeared that users with similar interests were more likely to share rules. I decided to investigate this further, as part of a study about how users customize software.

The manager who used the field customization features in the Information Lens to create a new form of message type for private communication between himself and his secretary (Mackay, 1988c) would probably be considered user innovation. In the latter case, the user used the software in ways that were unintended and unexpected by the software designers.

Four people shared rules or information about rules with others in the organization. The first two are not surprising. I, as the researcher, provided five sample rules for beginning users. Three rules were chosen because users in an earlier study had found them useful; two illustrated a trick that users might find helpful. The author of the local version of the software gave rule advice to approximately one third of the active Lens users, but did not provide copies for them. He introduced a number of innovations, but these were incorporated into the design of the software and made available to everyone.

In addition, two other people shared their rules with other members of the organization. One was a non-technical person who liked and used the locally-developed version of Lens.

He adopted a strategy, but not the actual rules, of a colleague. He then provided a complete copy of the rules to another non-technical colleague. The other person could be considered a "rule-guru": a highly technical person who wrote the most rules and provided feedback and ideas to the developers. One of the top technical members of the lab, he began experimenting with Lens as soon as it became available and created almost 100 rules. He created a number of innovations that were incorporated into the local version of the software.

Rule guru: "Lens gets more valuable as more people use it...so somebody has to start.

The rule guru created a number of complex, and useful rules that could be traced in other people's rulesets. I traced three of these rules or rule combinations: One allowed users to identify sets of related but uninteresting rules and delete them (the "boring" rule). Another pair identified jokes by a well-liked author and deleted jokes by others. Another identified a person from outside the lab who was announcing seminars and deleted the announcements without deleting personal messages from him. (Most people were uncomfortable with automatically deleting messages from a particular sender, even if the sender never sent anything of interest.)

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

One might have expected the developer to be the primary source of rules. After all, he has the most knowledge of the software. But in fact, he only provided advice, not sample

rules. His rules were designed to test the system and were quite different from those of a normal user. The other three rule-sharers were interested in how people used the system. They each developed rules that they thought would be useful for themselves, but also thought about their general usefulness. They each addressed a different need: "getting started", non-technical user, and programmer.

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

Users have a number of incentives to share, including taking advantage of each other's work, learning new methods of accomplishing useful tasks, avoiding errors, and generally saving time. The net result is that users adopt patterns of use from each other, propagating both useful innovations and errors throughout the organization. New users are more likely to customize, but they have less knowledge and skills than they will later. They are more likely to adopt others' patterns of working, even if they aren't suitable or contain errors and inefficiencies, because they do not have the skills or knowledge to evaluate the effectiveness of the software.

The function served by the latter group may be similar to the gatekeeper phenomenon 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, several people in the organization developed strategies for using rules, which they shared with their colleagues. However, the number of users was quite small, making it difficult to generalize from these results. These findings raise several questions. Do certain people identify themselves as a source of customizations for others? Are the exchanges one-way or mutually shared? What are the circumstances under which people decide to share their ideas and customizations? The second study will investigate these questions further, in the context of a study of how users customize software.

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

1. Interviews
2. organizational changes
3. trips
4. software conversion
5. earthquake

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 be an important component in the user's behavior. Although I do not have access to all possible relevant events, I have recorded several that may have influenced users. Table 3-I shows the dates when I visited the site and interviewed participants in the study. 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 user's perceptions of Lens, which affected their subsequent behavior. Not only did they 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.

Two people took advantage of events that provided extra time, such as a three-day weekend. Because trips usually corresponded to an increased need for mail handling capabilities, it was not surprising that people reported changing rules when planning for or returning from trips. In the example of user ***I***, described in 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. He commented "It changed my life.".

Problems with the system and the conversion from one version to another provide additional opportunities for people to re-evaluate their rules. In fact, all of the people who converted from version two to three changed the number of rules and seven of the eight also changed the types of rules. (The person who did not change simply recreated four of the six rules he had used before.) The effect of the conversion from version two to version three of Lens is shown in table 3-II.

Check this:

Notice that all ten users showed a general increase in the number of rules over time, within each phase. However, when they shifted from one version to the other, five decreased the total number of rules. (All of these people increased the number over the following weeks or months.) Many users commented that they rarely check to see whether their rules work, so ineffective rules tend not to get deleted. Thus, the conversion from one version to the next may have served an important function for these users, causing them to reevaluate all of their rules and begin with a new set.

Table 3-II: Patterns of Rule Changes From Version 2 to 3

Person	Version	Start	End	Range	Comments
A	Two:	3	14	3-14	Steady increase in rules. Changed only when the 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	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	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	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.
I	Two:				
	Three:	4	4	4	Recreated 4 rules. Accidently deleted 2 months later. Used autogetmail. Added 4 rules 2 months later.

Figures 3-3 through 3-7 show the a more detailed view 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 (the scale here 1 day per notch on the Y axis). For most people, the conversion occurred in January. However, one person converted in December, and several others stopped using version two 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.

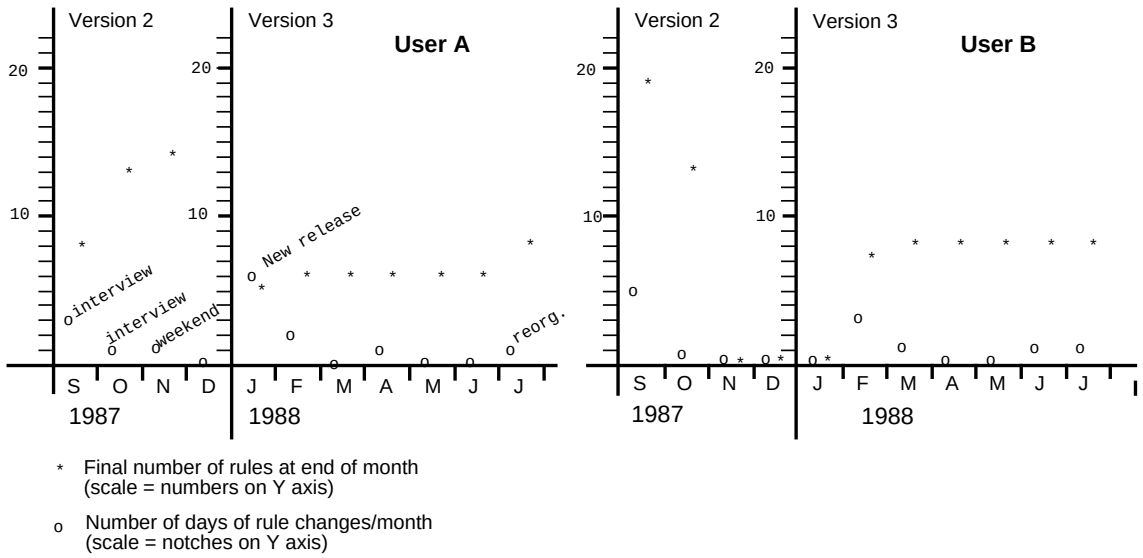


Figure 3-3: Conversion from version 2 to 3: Users A and B

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 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

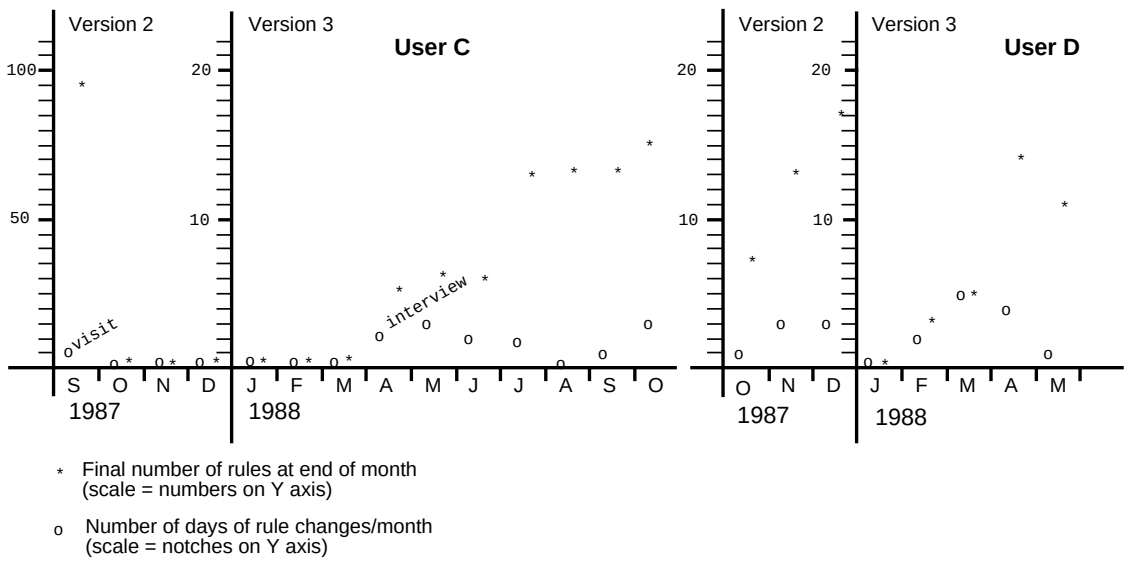


Figure 3-4: Conversion from version 2 to 3: Users C and D

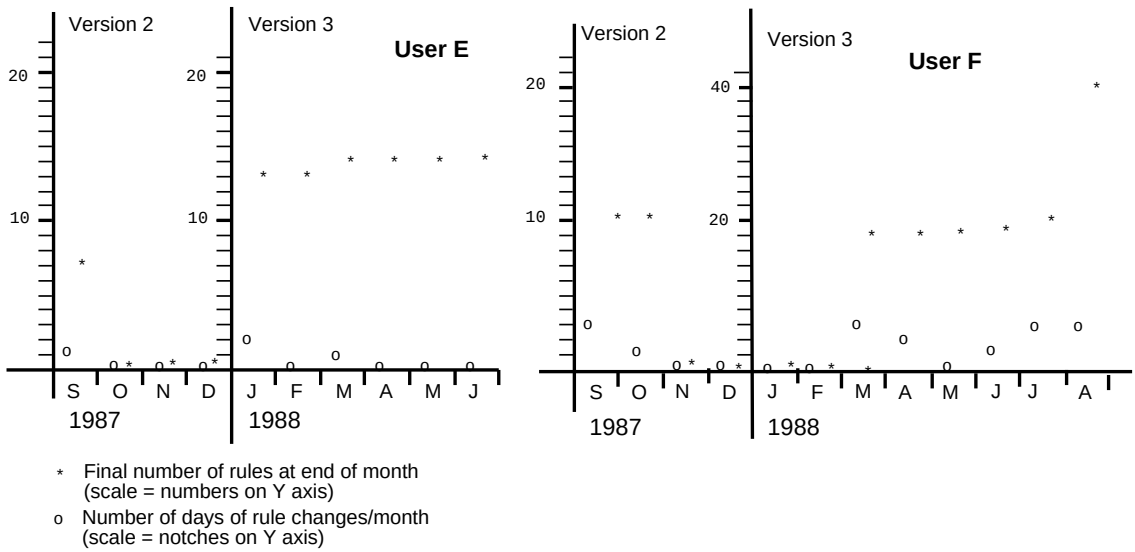


Figure 3-5: Conversion from version 2 to 3: Users E and F

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 D created 12 rules and then experimented with adding and deleting rules over the

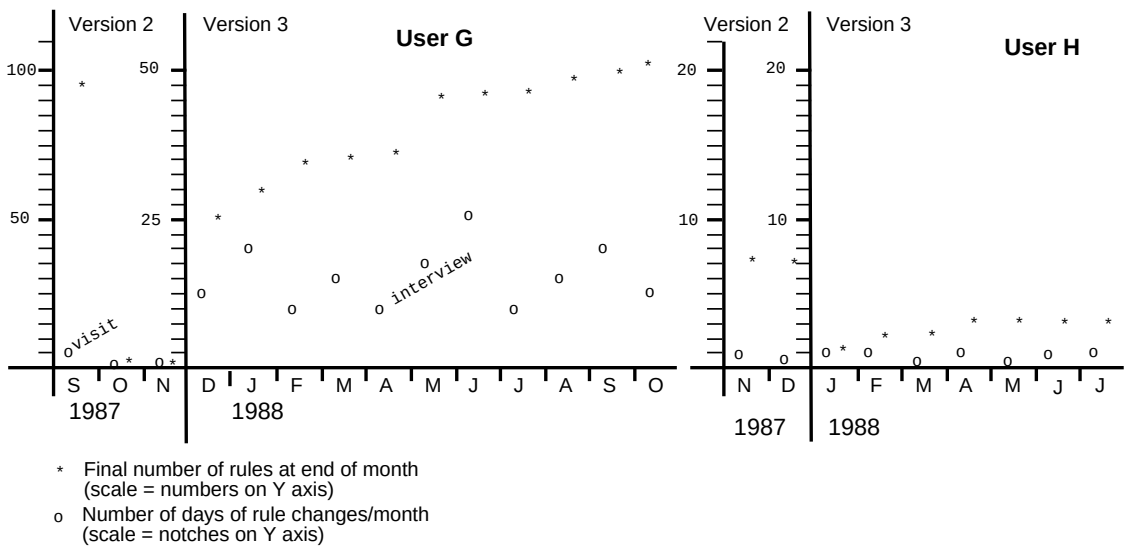


Figure 3-6: Conversion from version 2 to 3: Users G and H

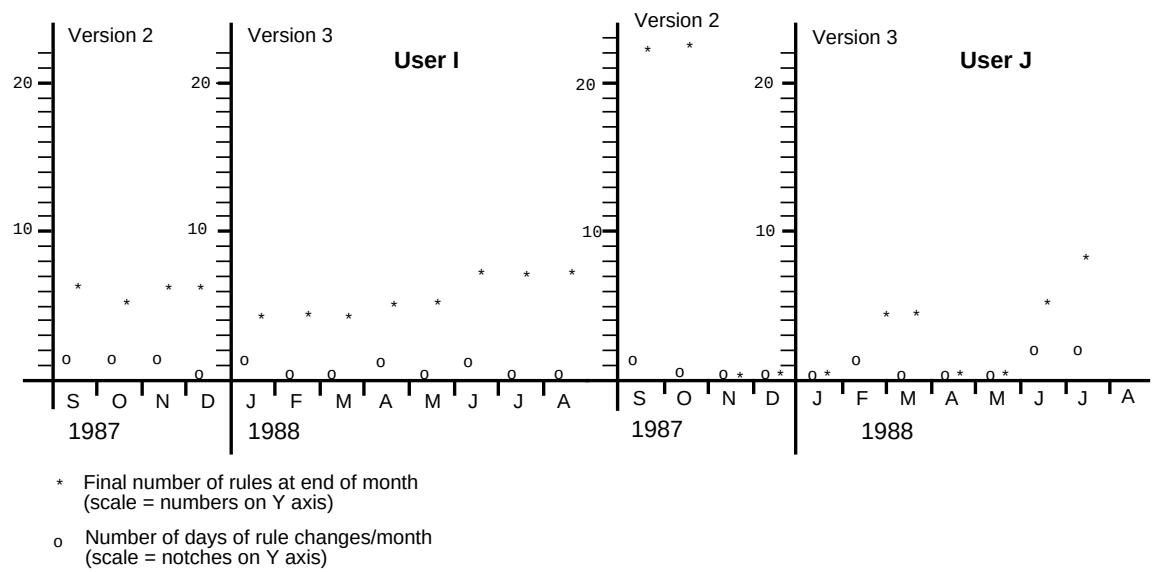


Figure 3-7: Conversion from version 2 to 3: Users I and J

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 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.

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.

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.

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 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 this time. He then steadily increased the numbers of rules, 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 3.3.2. He maintained these three rules without further change for the next six months.

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.

User J

3.3.8 Arrow *g*: Individual factors affect user's choices

1. Experience with LISP affects choice of rule editor
2. Changes in job
3. Word-of-mouth from others

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.

One secretary commented that "most of the stress in my job comes from email.". She was interested in using Lens to help her keep track of her messages (correlates with tasks in her job). Another person with a very large volume of mail that required constant attention decided it was too risky to use Lens: "I can't afford to be a guinea pig."

Changes in a user's experience with the job, such as the example described in section 3.3.2, effect how they operate.

The effect of Uncertainty

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 anything per se, but allow people to prioritize their work and get important/large things done. (If the input appears faster than the ability to do it, even if only in cycles, then this is necessary. cf "The twigging of overload".)

Sometimes this organization has to do with organizing the environment to remind people, or make certain things easy to find, or to place like projects together. (e.g. there's a cost to switching among tasks, so the organizing cost of placing like tasks together can produce an overall reduction in the time spent on tasks, even if the organizing costs time, because

of the reduced "startup time" for each task. This only works if the tasks appear in random order, rather than pre-categorized. Thus, a manager with a secretary who pre-categorizes work may save the manager time -- a different person in the organization pays the cost than the person who gets the benefits.

"Uncertainty" in work reduces people's incentives to spend what seems like "non-productive" time on organizing -- causing an overall decrease in productivity. If a project is "on hold" for an undetermined amount of time, if people are "about to move", etc., then behavior changes. People must make the trade-off. If you think you'll move soon, wait to do the org activity after the move, to make it more productive. If you think it will be a long time -- do the org behavior now and get on with things...act as the uncertainty isn't there. Could we measure productive/less productive people -- or people at different times in their work lives in more or less uncertain times and find out if there is an overall productivity cost? If so, is this really the reason?

Cost of not knowing that your office will be moved. Don't do some of the organizing activities that help day-to-day, because it is assumed that everything will change again soon. So you stay in a disorganized state for a long time -- since the state of uncertainty seems to usually extend past the predictions. (Mark Twain -- nothing so surprising as an event, long predicted, that finally comes.)

3.3.9 Arrow *h*: User's choices affect individual factors

1. Example of trip, later affects understanding/use of mail

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 that affect the individual user.

One of the interesting factors is that a message is not a static object with respect to a user. An individual will react differently to a message that has arrived but has not yet been read and one that has been read. One can think of the "lifecycle" of a message with respect to the user: It first appears as "new mail". Depending on the user's state (busy, tired, bored) and general disposition toward mail (like to read it, feel overwhelmed by it), this will be greated with greater or lesser enthusiasm. The next step is to see the header of the message in the browser. The reaction to the message will be influenced by a variety of factors, some dealing with the message itself (e.g. who it's from, what the subject line reads) and others independent of the message (e.g. how many other messages are in the browser and how "important" they are). The next step is to act on the message; deleting, printing or reading it. This decision is affected by many factors, such as length, perceived importance, and perceived timeliness. Once the message has been read, more decisions are made.

If the user decides that a reply is necessary, the sense of urgency will vary over time. Thus, if I read it 20 minutes ago, it's probably OK that I didn't reply. If I read it 3 days ago, I start to feel badly that I haven't answered yet. After 2 weeks, I generally begin my reply with an abject apology. After 3 months, I give up and it isn't important any more.

Thus, rules that deal only with the content of the message is unlikely to be fully satisfactory for some people, because their relationship to the message changes over time, even though the content of the message itself does not. However, users were very clever in dealing with the problem. Rather than specifying specific items about messages, they created and ran different sets of rules over the messages, depending on their current state. Several people created "vacation" rules, which were much stricter about deleting than normal rules.

Another observation from the lens study was 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.

When studying the introduction of the information lens, I observed several patterns in people's use of the software. They found that it was not "trivial" to use, and not just because of speed or overt user interface issues. The problems were related to the

difficulty involved in articulating behavior and the costs associated with organizing activities.

Some situations seem to increase the likelihood that people will spend time "organizing". Not surprisingly, when the researcher was actually there, people worked on their rules more. But the rate actually rose *before* we arrived; they remembered to use the system and collected questions for us to answer. People organize their offices and software environments to remind themselves of things (cite Malone's desk paper). (Cite example of different email strategies and management of inboxes in Mackay (1988).) So it's not surprising that being reminded about a tool that helps one organize will result in a higher probability that the tool will be used.

3.3.10 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 idea for the Information Lens was grounded in the growing needs of people to manage an increasing volume of electronic mail. As a member of a leading edge research community, Malone was exposed to the technology at an early stage in its evolution and was able to observe the problems that would later face people other organizations that rely heavily on electronic mail for internal communication.

The specific design of the solution was deeply influenced by the computer and organizational environment in which it was created. For example, the notion of the "anyone" server is interesting to an organization that conducts all of it's "broadcast" message traffic through distribution lists, rather than through electronic conferences, such as Netnews.

When the technology was first introduced, it failed miserably. Not only was it buggy and slow, but it also violated a number of principals in terms of how people used electronic mail. The next version worked better, and several people used it. One person found a "debugging" feature that allowed him to run rules at a different time. This addressed a problem and caused a number of people to start using it who had refused to do so before. Then, a local programmer built a version that fit with the model that everyone was used to. This version made the "bug" discovered earlier into a useful feature. At this point, a relatively large number of people started using lens, and used it in ways quite different from that predicted in the original design.

New feature When creating multiple rulesets, it becomes important to indicate when the ruleset stops. Similar to writing a program. So the developer of version 3 introduced the "stop" action. At the same time, he allowed for multiple actions. People could now move a file to more than one folder, or forward it, file it and mark it as "replied to".

Site's influence on design

The original design was influenced by and derived from a set of needs at the original site. As is often the case (cite Tatar), the implementation made a number of assumptions and made a number of things that people took for granted harder, while making other things easier. (Even though it was designed to be purely an addition.) Thus, the use of the NEXT key was different, and caused people many problems. This of course is only a problem for this audience, which has a set of expectations based on their use of other related systems. However, it is almost always the case that there are interaction effects with other technology that is simultaneously available on the system.

What about the costs of using the new technology? People tend to think about the

benefits of introducing a technology, but not about the extra burdens it creates. some of these burdens are quite small -- eg. looking through a large message type tree to select a message to send, rather than simply sending it -- but they affect the users' perception of the system.

The next version of the software was better accepted, and used in different ways. This new use suggested a set of different possibilities. Interestingly, this lead to multiple development paths for new versions -- object lens and argus.

Reasons for developing own version (three): versions one, two were slow, buggy, characteristics like "next" didn't work and were expected to. No examples, too complex. Didn't know what to use it for. (But SL made 98 rules.)

Ideas from study changed ideas of what's available / possible. Researcher worked with RR - new ideas. Also, simply talking to people about use of email changed ideas. And -- people changed views when told of idea of lens.

This study was interesting because the software developers were very close to the users of the software and changes in use were fed back in interesting ways. One of the effects of this was that the users taught the developers about what Lens was -- rather than creating a system that is limited to duplicating the nominal secretary.

Lens can be examined from different perspectives, each of which can be demonstrated to have changed over time. The original developers of Lens designed it meet a particular set of perceived needs and generalized it to meet anticipated needs. 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 prioritzed set to the

manager (or Lens user). People who listened to this had different reactions. Some assumed that this would be very useful. Others recoiled and refused to use it. "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.) Some people actually used (and continue to use) the software in this way. However, it quickly became apparent that many users wanted the prioritizing capabilities -- but only *after* they had read their mail. This was possible through a debugging option in the original software, and became a key feature of the third version of the software.

Through observation of actual use, the developers examined these needs and developed a new, richer understanding of the software.

Some of the changes were personal and did not affect the overall use of the software; others influenced the software designers and resulted in significant changes in subsequent versions of the software.

This diagram does not adequately illustrate the role of changes over time. Thus, as mentioned earlier, Lens is not a unitary phenomenon. It changes both through use and through on-going development. Thus, we should really think of the Lens box as a series of boxes with different changes over time. These changes will vary by individual, since different people use different features of the software.

Added multiple rulesets, boring rule.

Changes in the Technology

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".

Insert picture of Version 1 Rule Editor here

Figure 3-8: Original Information Lens Rule Editor: Version 1

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

Insert picture of Version 3 Rule Editor here

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

Version three also explicitly provided users with the ability to group personal rules into different *rulesets*, which can be executed at different times. Rules in the "new mail" ruleset are applied automatically when new messages are retrieved to the user's workstation, before the user sees the new messages. Rules in other rulesets ("folder" rulesets) are

applied when the user explicitly invokes them on a specified folder. Users primarily invoke folder rulesets after they have at least skimmed the messages in a folder.

Two different, independent development efforts at MIT -- Argus and Object Lens. One focused on extensions based on user innovations, scaled prototype, integration into Unix environment, handling of information other than mail. Explores concept of annotation/filter pair -- to allow receivers as much control as senders (more). Defines problem as a balance of control between senders and receivers. (Results in researcher innovation) Object Lens explores issues related to object-oriented messages and active agents.

3.4 Conclusions

Different individuals react to the same technology in different ways. Each person uses some features and disregards others; two people may use completely non-overlapping sets of features. Previous experience with other software affects each user's expectations about what Lens is and can do, which affects the user's subsequent patterns of interaction with the software. All of these factors encourage diversity in the use and perception of the technology. Through observation of actual use, the developers re-examine their own understanding of the software and develop a richer understanding of both the users' needs and the role of the technology in assisting with those needs.

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. Some users are likely to share their customizations and a few may actively create and distribute customization files for other people's use. Will different people act in this "sharing" role (get a better term, like gatekeeper) for different software applications? Or will the same people share their files across applications. (The Lens study found that individuals sorted themselves into

different areas of expertise. Other users came to know who was knowledgeable about which area and would tend to ask that person for help and forward interesting information to that person. Customization files may be that way too.)

One of the most interesting phenomena at the organizational level is the potential for the creation of "customization experts". They often share their customizations with newcomers or people who don't want to learn how to customize (based on the principal that it's more convenient to copy than create). This spreads usage norms throughout the organization and affects relationships among people. Innovations by people who do not share their customizations will not be as effective as those of the customization "gatekeepers".

These patterns of sharing are similar to those found by Allen (1972) , who identified "gatekeepers" in engineering organizations. Gatekeepers are highly skilled individual contributors who actively seek technical information outside of the organization translate it for the use of internal colleagues. (Notes: Trigg's term: bridge, not gatekeep. My findings: some organizations can have everyone be a gatekeeper. Similar to what academic researchers do -- they evaluate their own skills and interests, and then look for holes in the field or new opportunities, and try to become the unique expert in that area. Technical staff at PARC did this -- via choice of distribution lists, sending FYI info to each other and becoming expert in certain areas.)

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

3.4.1 Implications For Software Design

Software manufacturers, in response to customer demand, are increasingly making software applications customizable by end users. However, the results of this study indicate that customizable software is not always used in the manner in which it was intended. Software design decisions will affect how individuals customize and how these customizations are shared within an organization.

Individuals make choices about when to customize, responding to external (or internal) triggers or deciding that the barriers are too great. The design of the software can affect these triggers and barriers. For example, reflective software.

The design of the mechanism by which customization can occur is important. Some customizable software, such as that on an Apple MacIntosh, has a "direct manipulation" interface. Users modify the software, from the default values provided by the software manufacturer. However, they do not have an easy way of sharing parts of their customizations. In such cases, each user really does make a completely individualized set of customizations, strongly influenced by the design decisions of the manufacturer.

Reflective software provides users with feedback about what they actually do. This is somewhat different from Zuboff's (Zuboff, 1988) notion of "informating", which provides information about the *system*. The notion here is that users need to know what their individual patterns of using the software are, and better yet, if they are effective. Prototype software is sometimes instrumented such that researchers can examine the user's behavior. The recommendation here is to provide similar capabilities for users. (Note that providing all possible information, such as keystroke logs, is likely to be less helpful than a summary of small set of patterns.) Users are often unaware of their actual behavior

(Mackay, 1988d, Allen, 1972). Choices about the cost of customizing are based on perceptions, not actual behavior. It is almost a truism in Experimental Psychology that one should not simply ask subjects what they do, since they are rarely correct.

Suggestions about reflective software should apply to learning, as should the methods of sharing customizations... The benefits of artifacts in customizations relate to learning as well -- history files, emacs logs, records of use in the context of the task -- may be shared, commented on by others (either other peers in the network or by software agents), and reflected on by the user him/herself.

Software manufacturers do not explicitly support the sharing of customizations. Nor do they consider the effects such sharing may have on the use of their software over time. Because most use of customizations is not reflective (users don't usually examine which customizations are truly effective and which are not), the patterns of use do not necessarily improve over time. Another important consideration is the impact of delivering a poorly-conceived version of the software. Usually, poor decisions can be fixed with a new release. But here, users who get used to one pattern will be unlikely to switch to a new one, even if it is better. HOWEVER, the solution is not to simply release the software without examples of customizations. This simply gives the role of creating the first set of standards to someone in the end user organization, who is unlikely to have either the skills or the time to systematically test the possibilities and design a useful starting point for future customizations. (Interestingly, the "lead-users" are less likely to continue to use an inefficient pattern, since they tend to be more reflective about their own use of customizations. It's the people they gave their early versions to, and the people who received it from them, who are the ones who suffer.) Individual differences also play a part. Each individual has a different set of triggers and barriers that affect when and how much customization is done.

3.4.2 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 have strong effects on 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 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. 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 changes, both technical and non-technical, and sometimes proactively modify the system to produce real innovations.

This suggests several design implications. Customizable software is not always used in the manner in which it is intended. Software design decisions are important because they affect how individuals customize and how these customizations are shared within an organization. For example, customizable software with a "direct manipulation" interface may be easier to use but harder to share. Software designers should consider how to assist the people who become the translators or sources of customizations for their peers. The quality of the examples they create will affect the overall perception and use of the manufacturer’s software.

Software manufacturers should consider designing software to be *reflective*, allowing users to become more aware of their own patterns of use, and provide methods for evaluating effectiveness. Reflective software is somewhat different from Zuboff's (1988) notion of "informating", which provides users with information about the state of the *system*. Reflective software should increase the user's awareness of how they actually use the software. Techniques used to instrument software for feedback to user interface researchers may be useful here, particularly those that summarize behavior. (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 rather than regular users. However, given the influence of these people on the rest of the organization, it is important to help them reduce the number of inefficient or ineffective customizations and increase the creation and sharing of user innovations.

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

Individuals over a period of two years used electronic mail and a version of the Information

Lens. Individuals had highly divergent patterns of electronic mail use and highly divergent patterns of adopting the information lens. Individuals did not simply adopt the new technology, they changed it. They redefined the purposes for which it was used, some actually found ways to modify it, and they created innovative patterns of use. This is an example in which their behavior was both influenced by the introduction of a new technology, but they also influenced the technology itself.

Conclusion: customizations are often sources of user innovations. Users can apply these features in new ways.

Rather than treating the user interface as a single system, which individuals customize in unique ways, consider creating collections of customizations that are specific to common patterns of use. Individuals should then be permitted to make unique customizations within these common patterns.

Note: do not assume that "programmers" or "secretaries" adequately defines a pattern of use. As shown above, individuals have diverse patterns of use within job categories. The patterns should be influenced by whatever combination of factors is most relevant.

Chapter 4

Research Method

4.1 Proposed Research Method

I will conduct two studies of users of MIT's Project Athena. The first study will provide an in-depth examination of the customization behavior of a relatively small number (30-50) of people on the Athena staff over a period of four months. The second study will provide a longitudinal look at customization questions asked by a large number (from a pool of 11,000) of people from the MIT community over a period of 2 1/2 years.

The first set of data will include individual interviews, questionnaires, and records of customization files. The interviews will be open-ended, based on a variation of critical incident technique. The second set of data will consist of questions from Athena's On-Line-Consulting System database (over 30,000 questions) for the past two and a half years, which are amenable to both quantitative and qualitative analyses. I will isolate questions about customization of several different software packages (text editor, text formatter, window manager, etc.). I will also interview consultants to clarify the context of the questions and to verify the dates of relevant events (e.g. system upgrades).

4.2 Research Site: MIT's Project Athena

MIT's Project Athena was created as an "experiment in educational computing" sponsored 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 in the middle of the 6th 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 Information Systems structure.

4.2.1 Organizational Structure

Project Athena has undergone several organizational changes over the past six years. A major organizational change occurred in the middle of the study and the two top managers left. The organization from the arrival of the current director (September, 1988) through December, 1989 is shown in Figure 4-1. Figure 4-2 shows the results of the reorganization.

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. Most middle managers were not promoted to senior management positions. The latter position was a combination of two and the total number of senior managers was increased from two (both of whom are gone) to seven. Concurrent with the reorganization was a change in offices. The primary goal of the reorganization seemed to be to give the newly created senior managers individual window offices, with lower level managers given interior, individual offices.

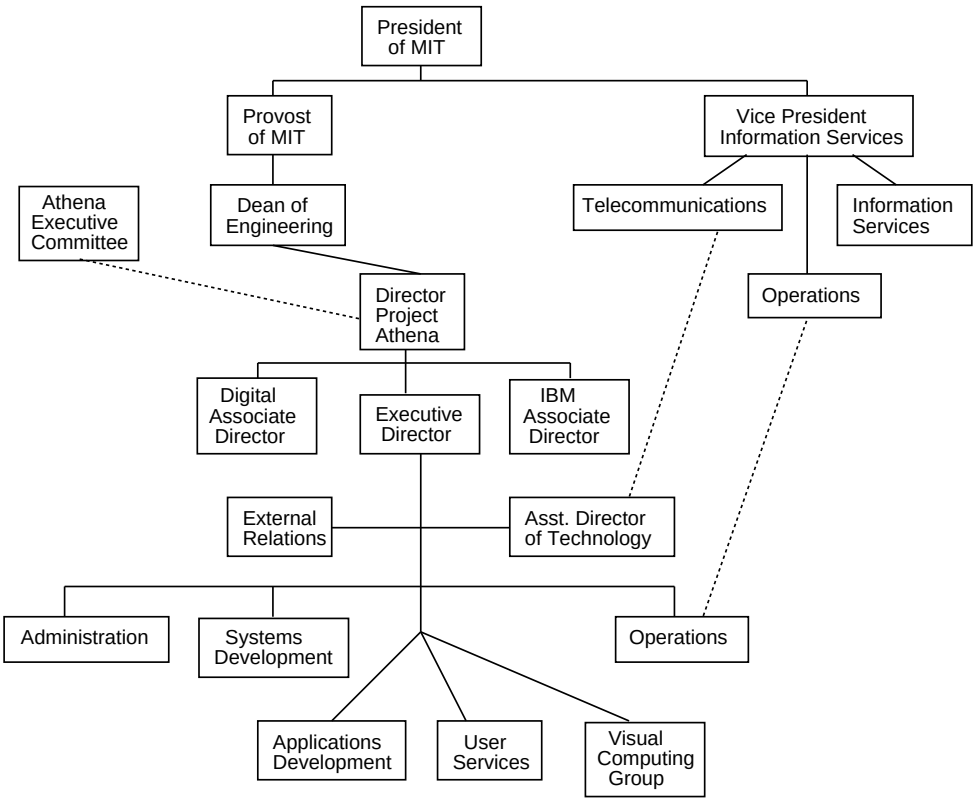


Figure 4-1: Project Athena: Organization through 1989

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

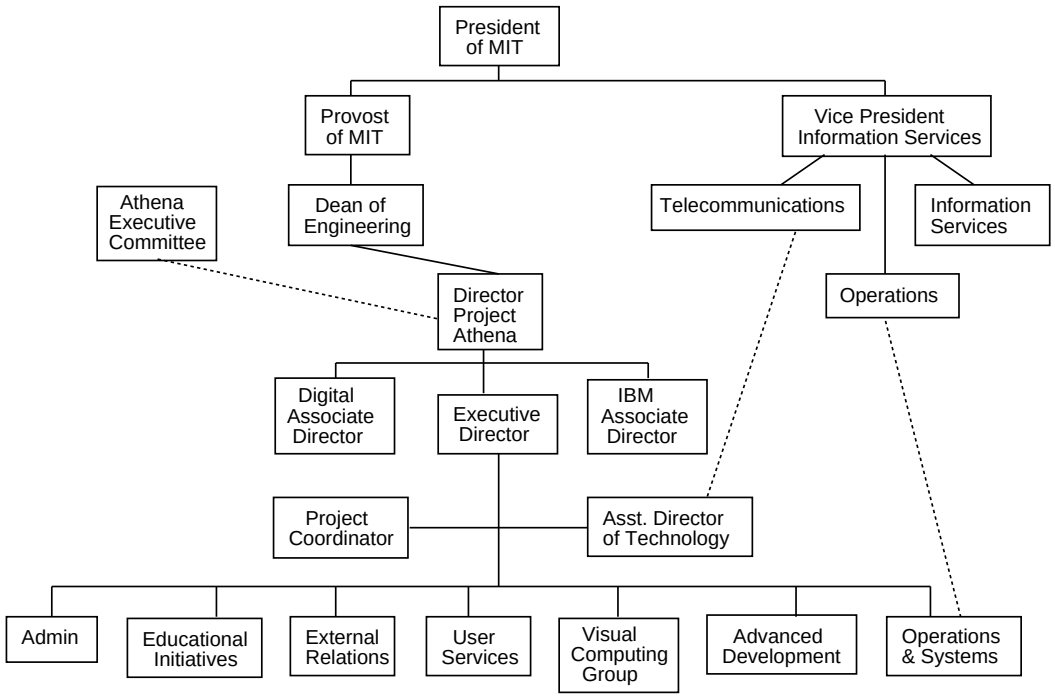


Figure 4-2: Project Athena: Reorganization in January 1990

university. In addition, some workstations have been placed in dormatories and fraternities and others are located in "departmental clusters".

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 pictures for the screen saver program) to functional (such as enabling the workstation to handle dial-in from home).

4.2.3 Software environment

Although initially unreliable, the Athena software environment is now quite stable, with major upgrades introduced at the beginning of each semester and at the beginning of the summer term. Users have individual accounts, can access the arpa net, 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, users can only access their own files. In 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) as:

> *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*, *awm*), mail systems (*mh*, *Xmh*, *rmail*), and *zephyr*, a system for displaying messages on user's screens. Other choices are available,

---

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

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 and flexibility. 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 interface, but offers limited customization. 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" which carries over to all (or almost all) of the software on the system, Unix and the X Window System provide users with many different forms of interaction with the system. However, 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, the 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 run 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.

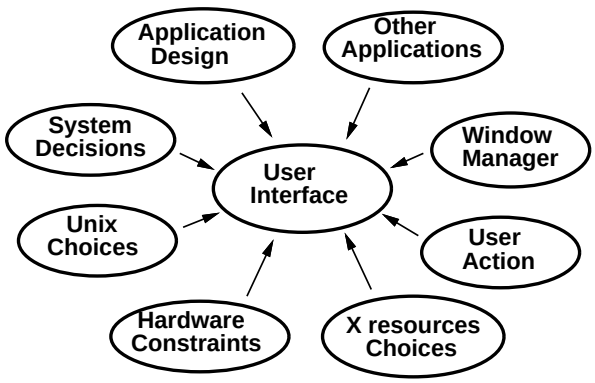


Figure 4-3: Factors that affect the user interface

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.[8]

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- shows what users may customize (or express preferences about) in the Athena environment:

Table 4-I: Customization Options

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

Some customizations are highly visible and might be noticed by someone walking by, such

[8]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 changes.

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.

4.2.3.1 On-Line Consulting System

When Project Athena was established in 1983, one of the services to be provided was a consulting service for users. At the time, the standard model of consulting was to provide a person to answer questions. However, Athena workstations were to be distributed across the campus in different "clusters". Providing full coverage by student consultants in each "cluster" would have required over 10% of the MIT student body, clearly impractical. Even if it were feasible, each consultant would have to act as expert in every area of the system. Beth Anderson, the manger of user consulting, addressed the problem by designing a distributed consulting system, which includes live consulting, a telephone service, and a program called OLC (Coppeto et. al., 1988).

The user types "OLC" at the Unix prompt and enters a question. A human consultant picks up the question (either immediately or as soon as a consultant is available). The consultant can send back a "stock answer" (i.e. an answer from a database of the most commonly asked questions), write a customized answer, or engage in a dialog with the user to clarify the question. If the user and the consultant are both logged on at the same time, they use a system called Zephyr, which pops up a window with text in it on the other person's screen. If either person logs out, the message is sent via electronic mail and will appear in the inbox when the person next logs in. Sometimes conversations continue over several days. If the consultant cannot answer the question, he or she will send it to someone else who has more expertise in the area. Approximately 5-10% of the questions

to the On-Line Consulting System are about how to customize software and workstations for private use.

4.2.3.2 Choices about Software Technology

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 a particular application or purchase it from an outside vendor, usually in response 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 no support is 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. The Student Information Processing Board (SIPB) is composed of students who like to program. They often contribute unsupported software that is sometimes better supported than the "official" software.

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. Students with interesting ideas will develop software and make it available 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. 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 tremendous internal resistance.)

Project Athena's 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 have the range of software available on most personal computers.

4.3 Study One: In-depth Examination of Athena staff

The purpose of the first 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-four times the number of staff. As "lead users" (VonHippel, 1986) of a new model of computing, they are the first to experience the problems and have the organizational power 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.

One might argue that generalizing from this group is difficult, both because of the unique constraints and also the unique control the group has on its use of technology. However, I contend that it is still important to study such groups. First, if such groups are, in fact, more innovative than other groups, it will be important to understand the characteristics that make them so. Second, the members of the staff are, for the most part, not programmers, and thus it will be interested to see if they are more empowered by the technology than similar people in other groups.

(Add to this section...)

4.3.1 Participants

The Project Athena staff consists of over 80 people, including managers, secretaries, technical and non-technical staff. Volunteers for the study will include a cross-section of managers, administrative personal and both technical and non-technical individual

contributors. The Athena staff provide a variety of services, similar to the MIS department of a large corporation. Two directors 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 have 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)

4.3.2 Procedure

I will interview a minimum of 30 members of the staff; my goal is to interview 40-50. I will also distribute questionnaires about customization use and communication patterns. For some participants, I will conduct additional informal interviews, based on opportunity.

Prior to each interview, I will ask participants to fill out a 2-page questionnaire with background information about the software they use, which they customize and how much, their programming background, and current job responsibilities. (See Appendix D.) I will also ask for:

1. a file with the dates of all system upgrades for that workstation,
2. a list of customization files and modification dates, ordered from most-recent to least-recent.
3. selected customization ("dot") files.

At a later time, depending on the participants' schedules, I will set up an appointment to discuss the questionnaire and ask open-ended questions about how the participant customizes his or her software environment. Interviews will be conducted for approximately one hour in each participant's office. I will begin by asking each participant to print out a second copy of the ordered list of customization files. (This will provide an indication of the rate of customizing and identify which files have been changed.) I will work from the set of questions shown in Appendix G.) As I continue to conduct interviews,

I expect to develop new questions based on issues raised in previous interviews. I will ask users to remember recent critical incidents in the past week. This strategy is based on Critical Incident Technique (See (Chapanis, 1969) for a discussion of the merits and problems of the critical-incident technique.).

I will ask to see selected customization files and ask the participant to describe the circumstances under which certain changes were made and to identify people who may have provided some or all of the customization file. I will conclude by asking people to fill out a questionnaire that identifies the people he or she talks to (See Appendix E.) and the sources and recipients of his or her customization files. (See Appendix F.) I will model these questionnaires after the sociogram devised by Allen (1972), for the purpose of identifying communication networks within an organization. Where possible, I will cross-check these answers with the people they identify and also examine the customization files themselves, which often includes a header with the author and the people who have subsequently modified it. Programmers are more likely than non-technical staff to follow this convention.

4.3.3 Proposed Analyses

The data collected are susceptible to both quantitative and qualitative analyses. The following are suggested methods of presentation of the data, with preliminary results included as examples where appropriate.

Descriptions of Individual Cases

I will select a small group of individuals and describe their customization activities in a case study format. I will choose individuals who describe strategies common to members of the group. I will also choose individuals who stand out as being unique in some way. In each

case, I will identify the person's role in the organization, technical skills, and place within the communication network. I will report how the person describes his or her customization strategy, factors that affect it, and the results of that strategy. It is important to note that, because customization decsions are based on *perceived* rather than *actual* costs and benefits, users' perceptions of their own behavior are important. Where possible, the perceptions will be compared to actual behavior in response to different external events.

Geertz (1973) (Geertz, 1973) distinguishes between "thin" and "thick" description. Thin descriptions report observations purely descriptively, primarily reporting acts devoid of meaning. Thick descriptions report observations imbued with interpretation, cast in the theory and language of the researcher. I will distinguish between each style of description.

Tables of questionnaire results

Table 4-I summarizes how people customize and share their window manager files. Individuals are listed by job type and ranked according to general technical level (high, medium, low). Within these subgroups, individuals are ordered by level of customization (ranging from 3, extensive customization, to 0, no customization) and then by years at Athena.

Network Graph

Figure 4-4 shows a preliminary map of who shares customization files with whom at Athena. The direction of the arrow indicates who was the source and who received the some or all of a customization file. Double-headed arrows indicate an exchange of customization files. The map illustrates two different kinds of customization sharers:

Table 4-II: Preliminary Results from Customization Questionnaire

Person Code	Job Type	Tech. Level	Years Here?	Use MWM?	Edit Back?	Customize Level?	Source of dot Files?	Share Files?
A1	MG	High	3	No	Revert	3	I wrote	Expert
A2	MG	Med	6	Yes	No	1	Auto copy	No
A3	MG	Med	3.5	Yes	No	1	Friend wrote	No
A4	MG	Med	1	Yes	Revert	1	Edited copy	No
A5	MG	Med	3	Yes	No	0	Auto copy	No
A6	MG	Med	3	Yes	No	0	Auto copy	No
A7	MG	Med	1.5	Yes	No	0	Auto copy	No
A8	MG	Med	0.5	Yes	No	0	Auto copy	No
A9	MG	Low	1.5	No	No	0	Auto copy	No
A10	MG	Low	3	Yes	No	0	Auto copy	No
B1	AS	Low	2	Yes	No	0	Friend wrote	No
B2	AS	Low	0.5	Yes	No	0	Friend wrote	No
C1	SP	High	1.5	Yes	Yes	3	I wrote it	Both
C2	SP	High	1.5	Yes	Yes	3	Edited copy	No
C3	SP	High	1	Yes	Revert	3	I wrote it	No
C4	SP	High	4.5	Yes	Revert	2	Edited copy	No
C5	SP	High	4	No	Revert	2	Edited copy	Expert
C6	SP	High	4	No	Revert	2	Edited copy	Expert
C7	SP	High	3.5	Yes	Yes	2	I wrote it	?
C8	SP	High	3	No	Revert	2	Edited copy	?
C9	SP	High	2.5	No	No	2	Edited copy	?
C10	SP	High	2	Yes	Yes	2	Edited copy	No
C11	SP	High	4	No	Revert	1	Edited copy	No
C12	SP	High	2.5	No	Revert	1	Edited copy	No
C13	SP	High	2	No	No	1	Edited copy	Expert
C14	SP	High	2	No	Revert	1	Edited copy	No
C15	SP	High	1.5	No		1	Edited copy	No
C16	SP	High	0.5	Yes	Yes	1	Edited copy	No
C17	SP	High	3	Yes	No	0	Auto copy	No
C18	SP	High	New	Yes	No	0	Friend wrote	No
D1	AP	Med	0.5	No	Yes	3	Edited copy	No
D2	AP	Med	4	No	Yes	2	Edited copy	Translator
D3	AP	Med	4	Yes	Yes	2	Edited copy	?
D4	AP	Med	6	Yes	Revert	1	Edited copy	No
D5	AP	Med	3	No		1	Edited copy	No
D6	AP	Med	1	Yes	No	1	Edited copy	No
D7	AP	Med	0.5	Yes	Revert	1	Edited copy	No
D8	AP	Med	0.5	Yes		1	Edited copy	No
E1	OS	Med	4	No	Revert	2	Edited copy	Translator
E2	OS	Med	2	Yes	Yes	2	Edited copy	Translator
E3	OS	Med	4	Yes	Revert	1	Edited copy	No
E4	OS	Med	2	Yes		1	Edited copy	No
E5	OS	Low	0.5	No	Revert	1	Edited copy	No
E6	OS	Low	5	Yes	No	0	Auto copy	No
E7	OS	Low	2	No	Revert	0	Auto copy	No
E8	OS	Low	New	Yes	No	0	Friend wrote	No
	MG Manager AS Admin staff SP Sys Programmer AP App Programmer OS Other staff			MWM: the Motif Window Manager	Revert: Customized new window manager to be like old WM	3 Programmed completely 2 Major edits 1 Minor edits 0 No edits		

technical experts who create files and broadcast them without feedback about their

subsequent use and "translators" who identify needs of non-technical or "busy" members of the organization and create and share customization files to meet those needs.

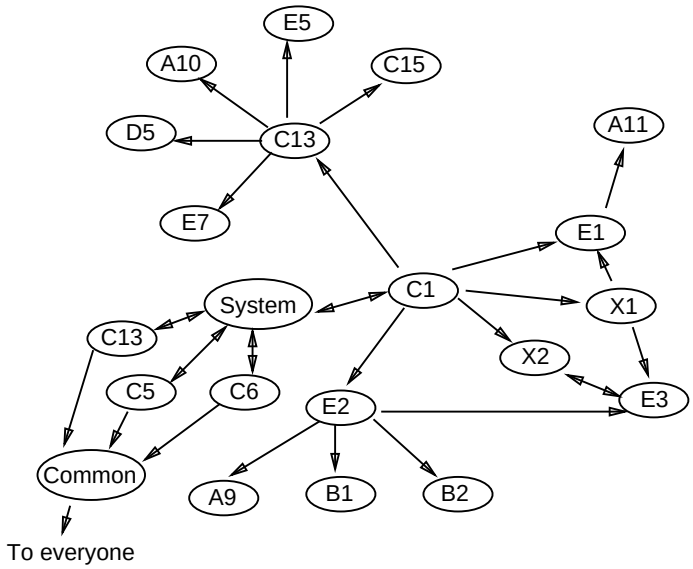


Figure 4-4: Exchange of customization files at Project Athena

Statistical analyses

The sample size is sufficiently large that certain questions may be amenable to a statistical analysis, although this will not be my primary method of reporting this data. At the simplest level, I will compare customization activities by job category and technical skill. I will also develop a coding scheme for level of complexity in customization files, to determine the relationship between technical ability and complexity. I do not have good measures of performance, except through self-report of level of overload experienced by participants.

Structurational analysis

I will relate the development of the Project Athena technology and organizational culture

over a period of 3-4 years, in a manner similar to the discussion of Lens in Chapter 3. I will relate different events to the structurational model.

4.4 Study Two: Longitudinal Study of MIT Community

Project Athena has an on-line consulting service in which users can ask the consulting staff for help on a variety of topics. One popular type of question in the OLC database (5-10%) is how to customize software. I will examine the kinds of customization questions asked and the circumstances under which they occur.

Re: OLC data: Predictions about results. I expect to see two competing processes: if the change in the software causes total confusion, then the users will be LESS likely to customize and will go into survival mode. If the change in the software makes them aware of the ability to customize, and doesn't overload them, then I expect to see MORE customization. I should be able to differentiate these by looking at the individual questions and by looking at the different types of changes. I can also track questions by user -- and users who ask more than one question -- and check if they're confused and upset by the new release. Will an external event that affects the system increase overall levels of customization?

OLC is interesting for this study both because of the technology itself (it is a technology that both resulted from and influenced the technology at Athena) and because it records the critical issues for users over a period of several years. I have access to the logs of all OLC questions for the past two years, and can correlate the questions to changes in the external environment (such as upgrades to the software or events in the school term). I plan to examine the questions themselves, to get a feel for the kinds of issues that

Another interesting aspect of the OLC logs is that they include dialogs between the

questioner and the consultant, and often include comments and suggestions by other consultants as well. I have access to the consulting staff and will ask them for their sense of the kinds of customization desired and the problems that people face.

The OLC logs are interesting because they contain more than just questions. They contain dialogs between users and consultants, as well as meta-dialogs among consultants. Other consultants may look at the interactions for a particular question and suggest other alternatives at a later date. This is important because it gives a sense of the difficulty of the question and the social fabric underlying the OLC consulting system itself. (See Appendix H for examples of OLC dialogs about customization.)

At the beginning of each semester, Project Athena introduces a new software release. The Fall ’89 release required users to become aware of "dot files", which users can use to customize different software applications and the operating system works. The Spring ’90 release will also have a user visible change, by introducing a new window manager (MWM). In addition, the system for customizing one application (Zephyr) will be completely revised. This will only be visible to users who already customize their use of Zephyr. Other applications will be unchanged.

4.4.1 Participants

Project Athena supports over 11,000 users on over 1,000 workstations situated around the campus. The consulting staff consists of five full-time staff and approximately 20 volunteers, who answer questions during consulting hours. During non-consulting hours, the questions are kept active and handled by the next available consultant.

Prior to the fall of 1989, Athena accounts were restricted to MIT undergraduates and faculty. Accounts are now available to all members of the MIT community, including

undergraduate and graduate students, faculty, MIT staff, research staff, Athena staff and visiting scholars.

4.4.2 Procedure

I have developed a set of Unix shell programs to extract and summarize questions from the OLC database. Preliminary sampling suggests that the number of questions specific to customization will not exceed 3,000 out of a pool of over 30,000 questions. For some purposes, simply identifying the questions and relating them to other events will be sufficient. For other purposes, a detailed reading of the questions will be important.

I plan to interview current and past consultants about the questions they have handled about consulting. (These people are members of the Athena staff and will participate in study one as well. These interviews will be in addition.) I will show them graphs of the OLC data and ask them if they can identify external events (e.g. a system change or major breakdown) that might explain the results. I will also examine the creation and use of "stock answers", which are created for commonly-asked questions.

4.4.3 Proposed Analyses

The dates of major changes in the system and major events in the school year will be plotted against the dates of customization questions, to see if any patterns emerge. The customization questions will be categorized according to topic and compared for differential sensitivity to external changes.

I will provide both qualitative and quantitative analyses. Because I have access to the text of the questions, which are actually dialogs between the user and one or more consultants, I will be able to clarify a number of issues that might not be accessible with just summary data.

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

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

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.

Coppeto, T., Anderson, B, Geer, D. and Treese, G. (February 1988). OLC: An On-Line Consulting System for Unix. *USEnix Conference Proceedings*. Dallas, Texas.

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.

Geertz, C. (1973). *The Interpretation of Cultures*. New York, New York: Basic Books.

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.

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. (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.

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. (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.

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.

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.

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.

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.

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

Proposed Thesis Outline

1. **Introduction:**
    a. *Why study customization?* Software designers increasingly add customization features to their software, yet how how people customize is poorly understood. Define customization and distinguish from user innovation.

    b. *Research Strategies for Studying Customization.* Discuss possible research strategies, and explain why the two main approaches are inadequate. Propose Orlikowski's model, enhanced with an individual-level co-adaptive analysis, as the theoretical basis. Justify based on findings from electronic mail study that demonstrated diverse appropriation of technology among individuals.

    c. *What are the expected findings?* The theoretical framework and observations from pilot studies suggest a number of predictions both about resistance to customization and the factors that encourage it.

2. **Theoretical Framework and Literature Review:**
    a. Describe and critique work in the area of information technology, including Zuboff, Winograd and Flores, Suchman, Burton and Brown.

    b. Identify two competing research strategies: the "technological imperative" and the "strategic choice" models.

    c. Describe the structurational model, as modified by Orlikowski and explain why it accounts for the data better.

    d. Describe co-adaptivity at an individual level of analysis and frame it in the context of the Orlikowski model. Explain why research in this area extends and enhances the theoretical model.

    e. Discuss factors that inhibit or encourage customization and the conditions under which users are proactive or reactive with respect to changing the technology.

f. Describe the user's decision process as a process of weighing the perceived costs and benefits against a user-defined threshold. Include the influence of the factors described above and how the factors may be weighted differently over time. Relate to the predictions identified in the introduction.

3. **Illustration of the Structurational Model Using the Information Lens:** Describe the evolution of the Information Lens in the context of the structurational model.
    a. *Site:* Describe the organization, hardware environment, and software environment.
    b. *Method:* Describe the participants, the data and the data collection procedures. (Data: open-ended interviews using critical incidents, and logs of individual rule changes over a period of up to eighteen months.)
    c. *Story:* Describe how the Information Lens was both affected by the organizations from which it came and influenced the behavior of people within the organization. Explain why an analysis of the diversity of individual adaptations to the technology is useful.
    d. *Individual analysis:* Describe how Lens caused users to modify their strategies for managing mail and how they appropriated Lens for their own purposes. Analyze individual usage patterns and how they were influenced by external factors, organizational factors, and the technology.
    e. *Patterns of sharing analysis:* Explain the relationships among participants in the study and how they exchanged rules among themselves.
4. **Research Method For Study at Project Athena:** Describe the research methodology and the types of data to be collected at MIT's Project Athena.
    a. *Site:* Describe the organization, hardware environment, and software environment.
    b. *Method:* Describe the participants (Athena staff and MIT community),

the data (interviews, customization files, and OLC logs) and the data collection procedures.

c. *Story:* Describe and analyze the evolution of Project Athena at MIT, including how Athena software was shaped by MIT culture and how MIT culture shaped the Athena software environment. Relate to the Orlikowski model, with emphasis on individual factors.

5. **Analysis of individual use:** Analyze individual patterns of customization by Project Athena staff, during a transition period in the organization and in the software. Identify what people actually do and then identify the factors that influence customization:
   a. *Triggers:* Identify the factors that encourage customization.
   b. *Barriers:* Identify the factors that inhibit customization.
   c. *Costs, Benefits and Thresholds:* Analyze the choices people make based on the perceived costs and benefits of user customization, and the factors that cause the choice threshold to change over time.
6. **Analysis of Patterns of sharing:** Identify the patterns of sharing among individuals on the staff and in the larger MIT community. Identify the factors that encourage or inhibit sharing.
7. **Discussion:** Relate findings to the theoretical model and discuss how the co-adaptive framework affects the software design process. Suggest directions for further research and implications for:
   a. organizational innovation and the adoption of new technologies,
   b. software design, including features that facilitate user reflection on own behavior and the sharing of customizations,
   c. the software design process, from iterative design strategies to processes for incorporating patterns of use and user innovations into subsequent software releases.

Summary of the characteristics of this research:

Topic:	User customization of software
Theoretical Framework:	Structurational model
Theoretical Perspective:	Co-adaptive systems (proactive and reactive)
Proposed Analyses:	Individual Triggers and barriers to customization Costs/benefits/thresholds Patterns of customization exchange
Sites:	Two "lead users" of distributed workstations.
Data:	Open-ended interviews. Logs of customizations. Historical records. Database of questions.

Appendix B

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 C

Ratings of Information Lens Features

1. Which of the following, if any, have been problems for you with Lens?

	Not a problem 1	2	3	4	Major Problem 5	Not applicable
Speed						
Running Rules						
Composing Messages						
Size						
Bugs						
Support						
Ease of use						
Format of displays						
Other						

2. How useful are the following features of Lens?

	Very Useful 1	2	3	4	Not useful 5	Never Used
Message Types						
Custom alternatives						
Pop-up alternatives						
Folder tree						
Sorting						
pre-reading						
post-reading						
within a folder						
Other rule actions						
Rule templates						
Anyone server						
Autogetmail						

Appendix D

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 [ ] Never programmed
   [ ] 1 - 2 programming courses [ ] Programmed high-level languages (hypercard)
   [ ] 3 - 5 programming courses [ ] Programmed for own purposes, e.g. research
   [ ] Computer science minor [ ] Applications programmer job (years_____)
   [ ] Computer science major [ ] Systems programmer job (years_____)
   [ ] Computer science grad. school

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

6. How often do you use your workstation?

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

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 F

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 G

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 H

Sample OLC Log files

Note: User's and consultant's names and usernames have been disguised.

Question Number: 2001

```
[2001] olc@ATHENA.MIT.EDU OLC-X_Window_System 09/03/89 14:10

Subject: example mwmrc files

Log Initiated for user XXX [0] (xx@yyy.MIT.EDU) [89/09/03 14:04:42]

Topic:          documentation

Question:

Short of track down mwm man file and printing it out, where can
I get printed information on how to cusimise mwm
_____________________________________________________________

--- Question grabbed by consultant aaaa@PORTNOY.MIT.EDU.
    [89/09/03 14:05:01]

--- Topic changed to 'xwindows' by aaaa@PORTNOY.MIT.EDU
    [89/09/03 14:05:05]

*** Reply from consultant aaaa@PORTNOY.MIT.EDU
    [89/09/03 14:05:44]

There is a default .mwmrc file in /usr/lib/X11/system.mwmrc.  Also,
there are a whole bunch of sample mwm resources and mwmrc files in
/afs/athena/mwm_samples and its subdirectories.

--- User xxxxx is done with question.
    [89/09/03 14:09:47]

--- Resolved by aaaa@PORTNOY.MIT.EDU.
    [89/09/03 14:10:10]

--- Conversation terminated at 89/09/03 14:10:11
--- Title: example mwmrc files
--[2001]--
```

Question Number: 2002

```
[2002]  olc@ATHENA.MIT.EDU OLC-X_Window_System 09/03/89 18:13

Subject: xscreensaver bitmaps

Log Initiated for user XX [0] (xx@yyy.MIT.EDU) [89/09/03 17:03:28]

Topic:          unix

Question:

I've seen some people with funky pictures on their
xscreensaver boxes (you know, the one that bounces around and
says "I'll be back" or whatever.) One of the bitmaps was an
opus, another was the bat signal.  How can I put bitmaps in
my screensaver???

thanks
XX
_____________________________________________________________

--- Unable to contact user xxxxx. User logged out.
    [89/09/03 18:07:32]

--- Question grabbed by consultant aaaa@PORTNOY.MIT.EDU.
    [89/09/03 18:07:32]

*** Mail from consultant aa@PORTNOY.MIT.EDU [89/09/03 18:12:52]

To: xx@Athena.mit.edu
cc:
Subject: Your OLC question about unix
--------

  I am sorry we were not able to answer your question while
you were logged in.

  You asked, "I've seen some people with funky pictures on
their xscreensaver boxes (you know, the one that bounces
around and says "I'll be back" or whatever.) One of the
bitmaps was an opus, another was the bat signal. How can I
put bitmaps in my screensaver???"

  Well, first of all, you've got to find the bitmap to use.
You can create your own using the bitmap(1) program (see the
man page), but there are also a VERY large number of already
created bitmaps floating around Athena.  I have a personal
```

directory full of them in /mit/aaaa/bitmaps, and there is a directory of them in /usr/include/X11/bitmaps, and there's another directory of them in /usr/sipb/bitmaps.X11.

Once you know what bitmap you want to use, you can either specify it to xscreensaver on the command line or add it to your .Xresources file. My .Xresources file, for example, comtains:

```
xscreensaver*float.bitmap: /afs/athena.mit.edu/user/a/bitmaps/map
xscreensaver*icon.bitmap:  /afs/athena.mit.edu/user/a/bitmaps/map
```

You can find out the exact format by looking in /usr/sipb/lib/app-defaults/Xscreensaver.

I hope I have answered your question. If you need more help, please feel free to use OLC again!

--> Jane Roe
Project Athena Watchmaker
Student Information Processing Board (SIPB) member
Volunteer OLC Consultant
aaaa@Athena.MIT.EDU

--- Topic changed to 'xwindows' by aaaa@PORTNOY.MIT.EDU
[89/09/03 18:12:58]

--- Resolved by aaaa@PORTNOY.MIT.EDU.
[89/09/03 18:13:03]

--- Unable to contact user xxxxx. User logged out.
[89/09/03 18:13:05]

--- Conversation terminated at 89/09/03 18:13:05

--- Title: xscreensaver bitmaps
--[2002]--

Question Number: 2022

```
[2022] olc@ATHENA.MIT.EDU OLC-workstations 04/20/89 22:17

Subject: customizing

Log Initiated for user XXX (xxx@yyy.MIT.EDU) (89/04/20 21:43:58)

Topic:          workstations

Question:

I have just received a VS2000 workstation in my office.  I would
like to set it up so that I can rlogin to it from dial-up or
ringworld.  I have created an /etc/athena/activate.local and
a /etc/athena/de/activate.local both of which contain :

#!/bin/sh
/usr/athena/access_on

I was unable to rlogin today (connection closed by foreign
host) (maybe the wording was Connection refused).  Is there
something else I need to do?

_____________________________________________________________

--- Connected to consultant John F Doe (jdoe@FOUNTAIN.MIT.EDU)
    [89/04/20 22:03:50]

*** Reply from consultant jdoe@FOUNTAIN.MIT.EDU
    [89/04/20 22:04:37]

You need to copy /etc/rlogind to /etc.  (At the moment it is
a link to /srvd/etc/rlogind.)  Probably telnetd should also
be copied

--- User has logged out.
    [89/04/20 22:04:40]

*** Reply from consultant jdoe@FOUNTAIN.MIT.EDU
    [89/04/20 22:13:47]

To: xxx@Athena.MIT.EDU
cc:
Subject: Your OLC question about workstations
--------

You need to copy /etc/rlogind to /etc.  (At the moment it is
```

```
a link to /srvd/etc/rlogind, which isn't always there.)
Probably telnetd should also be copied.

    John Doe
    Athena User Consultant

--- Resolved by jdoe.
    [89/04/20 22:17:38]

--- Conversation terminated at 89/04/20 22:17:41

--- Title: customizing
--[2022]--
```

Appendix I

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:_____