

**Beyond Iterative Design:
User Innovation in Co-adaptive Systems**

Wendy E. Mackay

Technical Report EPC-1991-130

Copyright © Rank Xerox Ltd 1991.

Rank Xerox Research Centre
Cambridge Laboratory
61 Regent Street
Cambridge CB2 1AB

Tel:+44 1223 341500
Fax:+44 1223 341510

BEYOND ITERATIVE DESIGN: USER INNOVATION IN CO-ADAPTIVE SYSTEMS

Wendy E. Mackay

Rank Xerox EuroPARC
61 Regent Street
Cambridge, England CB2 1AB
mackay@europarc.xerox.com

ABSTRACT

User interface researchers advocate involving users in the design of computer software. Yet current techniques do not go far enough: they fail to account for users who innovate.

This paper suggests that software use is *co-adaptive*: users not only adapt to software, but they also adapt it for their own purposes as user innovations. This paper reports innovations from two research studies which significantly altered usage patterns within each organization, changed the designer's conception of the software, and provided a unique source of input for the next generation of the software.

KEYWORDS: User innovation, co-adaptive systems, iterative design, customization, tailorability

INTRODUCTION

Most user interface researchers advocate iterative approaches to designing software and include users throughout the design process (Card et. al, 1983, Norman & Draper, 1986). Some iterative design approaches (e.g. Chapanis, 1969) concentrate on setting up usability studies in a laboratory and ask users to try successive software prototypes. Users' reactions influence subsequent versions of the software. Summative evaluation (e.g. Dick, 1977) measures reactions to the finished software and is often used by software manufacturers to ensure that the software is of sufficiently high quality to ship. Some researchers (e.g. Suchman 1987) argue that the use of software is a situated activity and must be observed in the context of the user's daily work. Participatory design (e.g. Ehn, 1988) is one such

approach, in which software developers and researchers work together with users, who actively provide input into the software design. These strategies are all designed to improve software by making it easier for users to explore and learn, increasing user efficiency, and either reducing the likelihood of errors or to making breakdowns more easily repairable (Brown and Van Lehn, 1980).

User-oriented design strategies have unquestionably improved the quality of user interfaces. Unfortunately, they do little to discourage software manufacturers from treating user-oriented design as an altruistic activity that can be cut when deadlines or budgets get tight.

Co-adaptive Systems

I suggest that a more fruitful way to think about users' interactions with technology is as part of a *co-adaptive* system. Users respond to the introduction of new technology in two ways: they adapt to it by changing their behavior and they appropriate the technology, adapting to meet their needs. On rare occasions, user adaptations become useful innovations and are shared within an organization.

The choice of the term "co-adaptive" 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. Yet complex phenomena, such as the Earth's atmosphere, are now being examined as a combination of effects in which the physical nature of the atmosphere affects living organisms and at the same time these organisms make changes that affect the atmosphere. Studying

both effects together is now seen as essential to the understanding of the atmosphere and is called *co-evolution* (NCAR Annual Report, 1985). Similarly, the term *co-adaptive* emphasizes the dual nature of the use of technology: people both adapt to technology and adapt it to meet their needs. These processes occur over time and influence each other.

Successful user innovations are a specialized and relatively rare form of user adaptation. Because innovation is rare, it is difficult to study: how can we predict who will innovate and when? On the other hand, even a few innovations can have a tremendous impact, especially if they are shared across an organization. They can provide powerful new ways of using the software and can even completely redefine it. Studies of user innovations may provide important information to:

1. *HCI researchers* who can better understand aspects of human-computer interaction,
2. *organizational members* who can benefit directly from practical innovations,
3. *software designers* who can use innovations to improve software designs, and
4. *software manufacturers* who can take advantage of user innovations as an inexpensive source of new product ideas.

In this paper, I report findings from two research studies in which users adapted to successive versions of prototype software and created a series of user innovations. I conclude with a discussion of the process of innovation and how different people benefit.

STUDY #1: USER INNOVATIONS WITH LENS

The first example of user innovation occurred in the context of a two-year study of the use of the Information Lens. Lens is designed to help users filter and organize their electronic mail. Users create *If-Then* rules which usually identify strings in the *To:*, *CC:*, *From:* and *Subject:* fields of a message and perform operations such as deleting the message or moving it to a specified folder (Malone, 1987).

Design of the study

The research site is a large laboratory with approximately 60 people within a research center in a large American Corporation. The site was chosen because of its use of the hardware and software environment necessary for the operation of the Lens prototype and its extensive use of electronic mail. The site and the original study are described in greater detail in Mackay et. al. (1989) and Mackay (1990b).

The participants included a varying number of users (ranging from 7 to 23) of the Information Lens, examined over a period of two years. Data included automatically-recorded copies of rules and rule changes, a series of structured and open-ended interviews, answers to questionnaires, and random snapshots of electronic mail use.

How users changed the model of using Lens

Three versions of Lens were introduced over a period of two years. Version 1 was presented to users as a sort of “automatic secretary” that could pre-sort mail into prioritized groups of messages. Users could specify a set of “local” rules (a *ruleset*) that would be run on incoming mail messages, before the user read them. Users could also set up a “central” ruleset, which fired whenever interesting messages were sent to a central rule server. In order to make ensure that a particular rule works properly, Lens included a debugging feature that enabled users to run the local ruleset on a particular folder.

Some people liked the idea of Lens: “*My goal is to read as little [mail] as possible.*” However, some did not like the idea that Lens would process their mail before they had had a chance to see it: “*I don’t want automatic mail sorting because I don’t trust a formula for sorting -- I’m afraid it would get sorted and I wouldn’t look at it again.*” and “*I could never trust a machine to read my mail for me.*” (Note that new managers often have the same reaction to giving up control over their paper mail to their secretaries.) In the preliminary interviews, before trying Lens, individuals expressed strong and diverse opinions about different features. Those who did not like the idea of a system that processed messages before they had read them chose not to participate in the study.

Eventually, one user discovered that he could use the debugging feature to run rules *after* reading mail in the inbox. He rewrote all the rules in his local ruleset. Rather than identifying the most important messages to read first, he sorted messages he had already read and moved them into archival folders. Instead of running this ruleset before reading his mail, he would run it whenever he felt it was time to “clean up” his mail inbox. This was a novel idea. As one person commented: “*I didn’t think of reading mail and then sorting.*” A number of users heard about this way of using Lens and decided to join the study. Their participation was made to coincide with the release of Version 2 of Lens, which allowed users to explicitly specify when to run their rules.

Users of version 2 quickly divided into two groups. Some chose to run their rules before reading their mail. Their rulesets tended to prioritize messages and sort them according to importance for future action. Others chose to run their rules after reading their mail. Their rulesets became a useful way of archiving messages that had already been acted upon (Mackay et. al, 1989).

Version 2 had fewer features than Version 1 and no longer explicitly supported the central ruleset. However, one user discovered that he could create more than one ruleset and run them at different times. He then proceeded to create rulesets that performed both types of sorting strategies. Version 3 was designed to support explicitly the use of multiple local rulesets.

Table 1 shows how people’s rule sorting strategies changed when they moved from Version 2 to Version 3. The “Pre-sort” strategy triggers rules automatically when new mail arrives. The “Post-sort” strategy allows users to run rules manually after they have been read. Version 2 users were forced to make an explicit choice between prioritizing mail (pre-sort) and archiving it (post-sort). Users who transferred to version 3 had the option of using as many rulesets as they wished. 70% converted to using both kinds of rulesets. (One person created three kinds of rulesets). Of those who continued to use a single ruleset, two people continued to use the same strategy and one person switched strategies.

Table 1					
User	Total V2 Rulesets	Sorting Strategy	Total V3 Rulesets	Sorting Strategy	
A	1	Pre-sort	2	Both	
B	1	Pre-sort	2	Both	
C	1	Pre-sort	2	Both	
D	1	Pre-sort	2	Both	
E	1	Post-sort	3	Both	
F	1	Post-sort	2	Both	
G	1	Post-sort	2	Both	
H	1	Post-sort	1	Pre-sort	
I	1	Pre-sort	1	Pre-sort	
J	1	Post-sort	1	Post-sort	
		50%	Pre-sort	20%	Pre-sort
		50%	Post-sort	10%	Post-sort
				70%	Both

The multiple ruleset innovation spawned additional kinds of rule strategies. In addition to distinguishing between “pre-sort” and “post-sort”, users began to design rules that took their personal work context into account. The most common strategy was to create two kinds of prioritizing rules, one for normal days and a significantly stricter one for when the user returned from vacation.

This new use of multiple rulesets also influenced the software design. Rules could now call other rulesets, so it became important to identify the conditions under which a ruleset should stop processing a message. The developer of version 3 introduced the “stop” action. In order to avoid creating additional rules, he also allowed for multiple actions in the same rule. Thus a rule could now move a message to one or more folders, forward it, file it and mark it as “replied to”. The implementation made certain assumptions about how people work; making some things easier and others harder.

In summary, the first innovation took advantage of a little-used debugging feature to redefine when rules could be run. This led to the creation of a whole new category of rules. The second innovation was influenced by the first, because it demonstrated different uses for multiple rulesets. Rather than system-defined rulesets, local and central, users

could specify when to run their rulesets. They thus developed different sets of rules to reflect the changing contexts of their work.

These users did not simply react to a static technology; they actively changed it. They were influenced by how the software was presented to them, by their previous experiences with the existing mail system and other software, by their on-going experiences with Lens and by the experiences of others in the organization. For a few people, Lens fundamentally changed their mail processing strategies. Different users took advantage of different features; a few used features in ways that were not anticipated by the original software designers and were not supported in the functional specification.

Each new version incorporated changes made by users who “broke” the previous software model and caused new changes in the patterns of use throughout the organization. Such a tight iterative design loop is unusual, in which fundamentally different views of the software, as created by users, continue to be incorporated into subsequent versions of the software.

STUDY #2: USER INNOVATIONS WITH ZEPHYR

The second example of user innovation occurred in the context of a four-month study of the how users customize software in a distributed computing environment. The full study is described in greater detail in Mackay (1990a, 1990b).

Design of the study

The research site is a University computer support center, with a full-time staff of over 80 people. 51 participated in the study, with jobs ranging from secretary to manager to programmer to non-technical support staff. Approximately half of the participants have little or no programming experience. Data include automatically-collected records of customizations, snapshots of software environments collected at random intervals, a series of open-ended and structured interviews, and answers to questionnaires.

How users changed the model of using Zephyr

The most extensive user innovations relate to Zephyr, a message system originally designed to provide

system messages to users. Zephyr was created to imitate the kinds of messages that system managers of time-sharing systems send out when the system is about to go down or when file servers are getting full. Because users of this software may be located at any one of well over a thousand workstations, the standard Unix program for handling system messages is of little use.

Zephyr has dynamically-created ‘subscription’ lists. Whenever a user logs into a particular workstation, he or she is automatically subscribed to information sources about the local printer, file servers, etc. If any of them go down while the user is logged in, the user receives a Zephyr windowgram which pops up on the screen.

Users quickly appropriated Zephyr and began sending each other messages. They next figured out that they could use Zephyr to find out where people were physically located. Shortly thereafter, a feature was added to allow people to “hide” from such inquiries by others on the system.

The subscription lists were extended to be more like electronic mail distribution lists, and users could add themselves to the list. Zephyr lists tend to concentrate on highly transitory information because the messages are only received by people currently logged into the system. For example, one list is for people who like to go out for late-night pizza or other food. Sending an electronic mail message would be annoying, because the messages would persist long after the appointed time for the pizza.

Individual users invented interesting uses of Zephyr. For example, one user took advantage of the fact that incoming Zephyr can be made to “beep” when they arrive. He created an electronic alarm clock by sending a rapid sequence of 500 messages to himself at a specified time. Although the beep feature was originally intended to draw attention to the arrival of a single new message he successfully modified it to make a sufficiently noisy alarm to wake him from a sound sleep. He claims this design has the practical effect of a snooze alarm, because, *“I can’t remember how to turn it off when I wake up. But by the time I remember how to turn it off, I’m awake.”*

The first version of Zephyr placed messages in a single location on the screen. This works fine for a small number of messages. However, if the user

subscribes to a number of Zephyr subscription lists, it becomes difficult to find important messages. One user extended this further, by placing broadcast messages into a terminal window and allowing them to scroll by. He describes his innovation as follows:

I wanted to direct messages to different parts of the screen. I monitor many [Zephyr lists]. I get two kinds of messages: things I want to read and things to look at if I'm already looking on the screen. I run two windowgram clients and put one in an Xterm. The junk messages are there, and they just scroll away. This way, I can segregate based on importance."

Basically, the important messages pile up in a stack on his screen and he has to click on them to make them go away. The "junk" messages appear in a window and scroll away automatically. So the window always contains just the most recent messages and the others disappear. He says *"It's very useful -- otherwise I couldn't get the right level of information."* He felt this was *"a painful enough distinction [between junk mail and personal mail] to do that customization."*

This innovation influenced the Zephyr developer to add a feature that allows users to specify where on the screen to display messages. Many users used this feature to isolate personally-addressed messages and keep them separate from more general broadcast messages. Interestingly, this use matches the most highly-valued type of rule cited by Lens users. Mackay et al. (1989) reported that 85% of Lens users created rules to distinguish personally-addressed mail messages from those addressed to general distribution lists.

The most interesting user innovation was one that spread to a number of users. The University provides a 24-hour on-line consulting system for users to ask questions. However, sometimes it takes a while to get an answer, especially late at night. So, users created a new Zephyr distribution list, called *Help* which anyone can subscribe to. People with questions simply post a Zephyr message to *Help*. Anyone who is watching may send back an answer. Since only the people who are logged in and currently subscribed see the messages, nobody is bothered with out-of-date questions that pile up. Some questions provide a flurry of responses while others are more difficult

and may be referred to the more formal consulting system. Essentially, the users have developed an informal, self-regulating peer-to-peer, consulting system, which has become a preliminary filter for the more formal on-line consulting system.

Users' expectations of the informal help system differ from those of the official consulting service. In the former, users know that they'll only get an answer if someone who knows the answer happens to be logged in at the time and is interested in answering the question. Thus they know that if they don't get an answer relatively quickly, they should go to the more formal consulting system. Thus, they've developed a quick, low overhead method to try first and a more formal method to try if it doesn't work. The consulting group can prevent questions from clogging up the official queue by encouraging questions to be answered quickly though the Help system -- or answering those questions themselves. This user innovation has affected the consulting operation as well. They have shifted their priorities and now less experienced consulting staff use the Zephyr help system to learn and to help answer the easier questions. This helps reduce the load on the formal system.

In summary, users appropriated a system originally designed to handle system messages and created their own peer-to-peer consulting system. As in the Lens study, user-inspired innovations affected subsequent versions of the software, which resulted in new patterns of use.

DISCUSSION AND CONCLUSIONS

User-created innovations are important: they are based on real needs and provide practical solutions to problems that real people face in the everyday world. Yet despite their usefulness, most such innovations are lost because nobody is looking for them. Even the users who create them are often unaware of them. The person who does something new usually does it to solve a recurring problem (Mackay, 1991) and once incorporated, the user is likely to forget about it and become absorbed into their daily patterns of using the software.

Software programs represent 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 software as a unitary phenomenon, this practice has the unfortunate effect of implying that it is static. It encourages people to misunderstand the process by which the software itself is changed through design and use. These studies demonstrate that software use can be co-adaptive and that users may play an important role as a source of new design ideas.

Who innovates?

Even when specifically asked, most people do not think of themselves as creators of innovations. However, if asked in the context of actions they've recently performed, many people can identify one or two things they have done to solve a specific problem. Even so, only a few people have the time or the inclination to actively explore new ways of using the software. Sometimes, users are unaware of an innovation because it is based on a misunderstanding of the structure of the software. In such cases, it requires someone who is familiar with the expected use of the software to identify the innovation.

In both studies, innovators were most likely to be the *translators* (Mackay, 1990a) who translate technical information into a form that their fellow users can use and understand. These findings correspond with studies of technological gatekeepers (Allen, 1986). Perhaps translators innovate most because they actively try to bridge the gap between the design of the system and the patterns of its use. Translators often share their findings with their colleagues (Nardi & Miller, 1990, Maclean et. al., 1990) and thus their innovations are more likely to spread to other members of their group. However, innovations are rarely shared outside the group and are thus lost to the organization as a whole and the software designers who might benefit.

When do users innovate?

Users are most likely to explore new software when they first encounter it. They draw from past experience and customize software as a way to learn and establish future patterns of use (Mackay 1991). After experimenting for a short while, most users settle into a limited set of usage patterns. At this point, most users don't think about their use, they just "do it" (Winograd and Flores, 1986).

Most user innovations didn't occur during this "regular" use. Instead, innovations were sparked by interruptions that caused the user to reflect upon his or her use of the software. Interruptions include software upgrades that disrupt normal use patterns, system failures, job changes, and observation of other people's usage patterns.

Who benefits from examining user innovation?

Studies of user innovations may provide important information to:

1. *HCI researchers* who seek to better understand human-computer interaction. Examining software use as a co-adaptive phenomenon can help researchers find novel uses of software and thus help to challenge existing assumptions about both software and users.
2. *Organization members* who benefit directly from innovations. Simply letting end users know it is possible to innovate may increase the likelihood that they will. Managers should recognize that sharing occurs and encourage its effective use.
3. *Software developers* who can reinterpret their software based on new uses of it. Designers can learn from two orthogonal points of view: the system architecture and the dynamic process of use. User innovations may challenge basic assumptions about the purpose of the software and suggest further design innovations.
4. *Software manufacturers*: who can benefit from a relatively inexpensive source of new product ideas. Studying users is more than just an altruistic act. Learning from user innovations may become essential in order to stay competitive in a constantly changing and difficult marketplace.

SUMMARY

This paper argues that the use of software is a co-adaptive phenomenon, in which users not only adapt to new software, but also appropriate it and adapt it to their needs. One of the most interesting forms of adaptation is when users innovate.

The people most likely to innovate are those who perform a translation role, interpreting the structure of a software system into usage patterns that can be

shared with others in the group. Because researchers tend not to look for innovations, we don't know how common or rare they are. Even if quite rare, however, they provide an important element for understanding the role of technology in organizations. In addition, they may be used as a low-cost source of product ideas. Software manufacturers must stop treating field studies of users as an altruistic activity that can be cut when deadlines get tight and recognize that users can be an important source of information about the next generation of software.

ACKNOWLEDGEMENTS

I'd like to thank Wanda Orlikowski, Lucy Suchman, Tom Allen, Tom Malone, Marcie Tyre, Ramana Rao, Beth Anderson, and Bill Gaver for their comments and discussions about these ideas.

REFERENCES

- Allen, T. (1986). *Managing the Flow of Technology*. MIT Press: Cambridge, MA.
- Brown, J. and Van Lehn, K. (1980) Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science*, 4, pp. 379-426.
- Card, S., Moran, T., and Newell, A. (1983). *Human-Computer Interaction*. Hillsdale, N.J.: Erlbaum Associates.
- Chapanis, A. (1969) *Research Techniques in Human Engineering*. Baltimore, MD: Johns Hopkins Press.
- Dick, W. (1977) Summative evaluation. In L.J. Briggs (Ed.), *Instructional Design: Principles and Applications*. Englewood Cliffs, NJ: Educational Technology Publications.
- Ehn, P. (1988). *Work-oriented design of computer artifacts*. Stockholm: Arbetslivscentrum. Also Hillsdale, N.J.: Erlbaum.
- Mackay, W., Malone, T., 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, TX. NY: ACM Press.
- Mackay, W. (October 1990). Patterns of Sharing Customizable Software. *CSCW'90: Conference on Computer-Supported Cooperative Work*. Los Angeles, CA. NY: ACM Press.
- Mackay, W. (1990). *Users and Customizable Software: A Co-Adaptive Phenomenon*. Doctoral Dissertation, Massachusetts Institute of Technology: Cambridge, MA.
- Mackay, W. (1991) Triggers and barriers to customizing software. *CHI'91: Conference on Human-Computer Interaction*. New Orleans, LA: NY: ACM Press.
- Maclean, A., Carter, K., Lövstrand, L. and Moran, T. (April 1990). User-tailorable systems: Pressing the issues with buttons. *CHI'90 Conference on Human-Computer Interaction*. Seattle, WA. NY: ACM/SIGCHI.
- Malone, T. (1987). Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination. *ACM Transactions on Office Information Systems*. 5(2), pp. 115-131.
- Nardi, B. and Miller, J. (1990). Twinkling Lights and Nested Loops: Distributed problem solving and spreadsheet development. *CSCW'90: Conference on Computer-Supported Cooperative Work*. Los Angeles, CA. NY: ACM Press.
- NCAR (1985). National Center for Atmospheric Research Annual Report. Boulder, CO: University Corporation for Atmospheric Research.
- Norman, D. and Draper, S. (1986). *User Centered System Design*. Hillsdale, N.J.: Erlbaum Associates.
- Suchman, L. (1987). *Plans and Situated Action*. Cambridge, England: Cambridge University Press.
- Trigg, R., Moran, T. and Halasz, R. (1987). Adaptability and Tailorability in NoteCards. *Proceedings of Interact '87*. Stuttgart, Germany. Amsterdam: North Holland.
- Winograd, T. and Flores, F. (1986) *Understanding Computers and Cognition*. Norwood, NJ: Ablex.