Wendy E. Mackay
Massachusetts Institute of Technology
Mail: E40 366
1 Amherst Street
Cambridge, MA 02139

26 November 1989

Professor Eric Von Hippel
MIT Sloan School of Management
50 Memorial Drive
Cambridge, MA 02139

Eric:

Attached is a preliminary draft of my thesis proposal, as we discussed, and a proposed schedule for completion of the work. Please let me know what you think. Thanks.

Regards,

Wendy E. Mackay

Thesis Proposal

Wendy E. Mackay
26 November 1989

Research Issues

How can the user tailor the system to meet his or her own needs? How can the system tailor itself to adapt to what it perceives to be the user's needs?

Whenever users learn to use a tool or a program, they learn both what it can do, and also what it can't do. Researchers in software engineering try to create and understand the optimal primitives. In Lisp, it's a list. In SmallTalk, it's an object. In a relational database, it's a relation. These primitives are powerful and suggest a variety of new applications -- they generate possibilities that are based on the primitives. At the same time, they limit other kinds of possibilities. This is why artificial intelligence applications are almost always developed in Lisp, wherease real-time graphics applications are developed in languages like C. Trying to write AI programs in C is very difficult.

Assumptions are explicitly (and implicitly) imbedded into applications, and some of those assumptions are passed on to the user. In addition, the users bring their own assumptions about what the software should do, based on their own needs, past experience, and understanding of the software.

"Powerful abstractions" may provide users with new ways of thinking about things: for example, the use of a spreadsheet caused people to think about their financial data as something that provided real information. (reword). But, at the same time, it limits the user who wants to manipulate data in different ways.

It simply is not possible to have an unconstrained system. And if it were, it has the problems of forcing the user to do all of the thinking about how to structure the problem. If the software provides some of the structure and the user provides the rest, and it's a reasonable fit, then it will be viewed as successful. (This is why you don't want to give everyone binary code to write...)

Thus, there are assumptions imbedded into any piece of software (by the manufacturer) and assumptions made by the user about what the system can and cannot do. It's important to recognize that users will make assumptions, no matter what the manufacturer does. Sometimes those assumptions overlap with the manufacturers, sometimes they don't. Sometimes they overlap with the assumptions made by other users and sometimes they don't. These assumptions both suggest new possibilities and simultaneously limit the kinds of innovations likely to occur.

Why is it hard to predict how users will modify software?

Why is it hard to predict how users will modify software to suit their own needs?[1]

1. The user's assumptions about what is possible may be hard to detect. (Implicit not explicit

[1] *Modify* can mean either actually changing the code or using the code in an unanticipated way. Thus, using a spreadsheet for handling numbers in budgets is anticipated by the manufacturer. Using it to manage text-based research data is another.

knowledge)

2. The user's assumptions may be hard to change.
3. Users may be unable to articulate their needs in ways that map to potential solutions.
4. Manufacturers may be unable to articulate their assumptions or communicate them to users.

Why explicitly challenging Assumptions is likely to be useful.

The question is whether or not one can predict the ways in which users will modify software to meet their needs. If yes, big win, can provide user modifiable software that's pretty useful. If not, useful to know why.

Intuitions:

1. Observe difference between users' perception of their own behavior and their actual behavior.
2. Provide feedback to users about their behavior.
3. Ask users to predict their own behavior with a new system and compare to actual use.
4. Observe changes in use over time, particularly in response to crises.

1. Predict which behavior is likely to change.
2. Predict how it will change.
3. Predict which behavior is likely to stay.

Providing feedback loops is critical because it provides information to users about their own behavior. As a result, users may detect previously unnoticed inefficiencies and fix them. Feedback also provides clues about how successful different strategies are -- and which may be likely to change.

What you want to do is to induce the "aha" reaction -- to get the user to change the problem slightly so that other solutions suddenly become possible. Sometimes redefining the problem is sufficient to produce a solution.

Notion of feedback loops is very important -- also the idea that it's not a single exchange between a sender and a receiver but the exchanges that occur over time. (Just as in the classic prisoner's dilemma situation, in which the optimal behavior for a one-time game is different from that of a number of games over time.)

Observe or establish that: users need information about the success of their interactions. In fact, they learn by doing, and it forces them to come into contact with what it is that they need.

Eric's concern about who makes this available -- the whole issue of prototyping comes in.

Involves changes in the incentive structure of different senders and receivers of electronic mail.

Availability of change: Making change available, in and of itself, may have the effect of changing the way users think about the problem -- and therefore creating solutions (or improved performance). But, asking people to predict what they'll do won't necessarily tell you what you want to know. They may not do it (which is interesting of itself, if you can figure out why. Perhaps the short-term costs are too high -- get alot of self-deprecating "I'm lazy" sorts of explanations.) Also, they may try it and not find it effective.

(What do they do? Why is it ineffective? What is the assumption about the nature of the work that was wrong or was it the assumption about how the system would help that was wrong? OR, often the case with computer systems, did they neglect to take into account the unrealized advantages of the off-line or existing system which go away with the new "efficient" system. People often don't realize that they accomplish more than one thing with a particular system -- and that what may be perceived as an inefficiency along one dimension may be providing important benefits along another.)

It would be interesting if the differences between described behavior and actual behavior provided clues (predictive) as to what would be useful.

You want to think about whether the user is at a steady state to begin with or not...some are always trying to come up with new and better strategies for managing their time, while others don't bother. Would expect the former to be more likely to stumble across useful innovations.

How do we detect assumptions?

1. Experimental method: *look* for them.
2. User descriptions: to someone with a different set of assumptions.
3. Comparisions: systematic comparison of competing assumptions.
4. User feedback loops: user challenges own assumptions.
5. Accidents: what happens?
6. New assumptions: present them and see what happens.
7. Refusal to do something that seems beneficial.

Preliminary Work

Pilot Data

In the course of a study of the Information Lens, I conducted a series of interviews which consisted of a preliminary interview to describe the subject's mail use, a description of the Information Lens, a "set up" interview to set up an initial set of rules based on the user's stated needs, and subsequent interviews and actual usage data over a period of many months.

The users in this study used several, quite different, versions of the software. For the purposes of this study, however, it does not matter, since the real interest in how the perception of the usefulness of the system changed, regardless of specific features. (Gonna have to justify this one...)

The basic observations were as follows:

1. The act of describing the Information Lens affected users' perceptions of how they used

electronic mail.

2. Users do not have particularly accurate perceptions of mail volume, distribution list membership, etc. (Some were more aware than others however.) This might be a clue, by the way, for predicting how users will modify the system. If we look at where perceptions differ from actual usage, it may be a clue to what the user actually needs and where they may be more likely to innovate. Can draw on stories of programmers remembering the time that something was programmed quickly, rather than the two weeks it took to find a bug.
3. Users' predictions of their own use the system were sometimes (often?) inaccurate...especially over time. This has to be controlled for by changes in their jobs, etc. Thus, becoming a manager would certainly change one's use of email -- and old predictions might not hold.
4. Use of the system for some people changed over time -- some evolved, some changed rapidly, in response to a crisis or "punctuated equilibrium". (I had to put that in somewhere...)
5. Using rulesets at different times allowed users to add implicit information about their own work situation (context) to enhance the value of processing the messages.
6. Cite the story about Ramana and Tom's use of Lens to go over a large set of email rules...

Learning-by-doing

1. Why is it critical?
2. Can we enhance its effectiveness? (ie influence what is learned?)
3. Can we predict areas that will change? (if not precise changes)
4. Can we change the design of the software to enhance modifiability?

Comments - where does this go?

Because users are often unaware of their strategies and can't explain them. Ditto their needs -- they tend to talk about them in terms of the solution sets they know ought to be possible (e.g. a faster card reader) rather than in terms of technolgy with a completely different character and sets of assumptions (e.g. terminals).

Proposed Research Strategy

1. **Continued analysis of pilot data**
2. **Set up a study with Athena Lens/Pygmalion/Lens at DEC**

   A user study is currently under way to study message types and the anyone server. I can use the same environment to test these ideas without interferring (and perhaps enhancing) the other on-going work
3. **Other user modifiable programs: Muse? Widget Builder? EVA?**
4. **Research Data**

   Interview data: ask users what they do, what they want to do. Ask them to predict how they will use the system. Then measure what they actually do, continue to interview them

> to get insights into how they change what they do and how they perceive it. What kinds of changes do they make? What kinds of changes do they ask for? What kinds of events change their use of the system? (In the original Lens study, the jump between versions forced people to recreate their rules. Some people simply retyped their existing rules, others tried something new. Should we arbitrarily insert a change like this into the system?)

What are my intuitions? What do I think would help me predict? I'm somewhat worried that it simply isn't predictable -- that it depends greatly on what the user needs/wants at the time. Also, a change for one user may or may not be useful to another.

What do I do now? Am I aware of it? If so, could partitioning it differently cause it to be more useful?

Politics Get my pilot data back from Tom, figure out how to use the Lens study, without hurting Maya. Figure out how to work with Tom and not go crazy.

Proposed Schedule

Month:	Thesis work	Other commitments
March:		
	Proposal topic Form committee Human Subjects OK Pilot data	
3/30-4/1		Video Workshop
April:		
4/1-4/2		SIGGRAPH Committee VisiCom paper
4/15		HCI paper Letter to Media Lab
May:		
5/1-5/8		CHI ’89 conference CACM paper due
June:		
6/2-6/3		SIGGRAPH Committee
July:		
7/25-7/31		SIGGRAPH ’89 conference
August:		
8/1-8/4		SIGGRAPH ’89 conference Media Lab talk
September:		
9/18-9/22		HCI ’89 Conference
9/17-9/21		Visicom ’90 ?
October:		
November:		
December:		

Thesis defense

January:

Discussion Topics

1. Tom Malone Discussion
    a. "Territory" dispute
    b. OSF proposal
    c. Athena Lens
    d. DEC funding
2. Obtaining data
    a. Pilot data from PARC study
    b. Human subjects committee OK
    c. Lens User Study
    d. SIGGRAPH study
3. Future career
    a. Andy Lippman discussion
    b. Geoff Bock discussion
    c. George Champine discussion