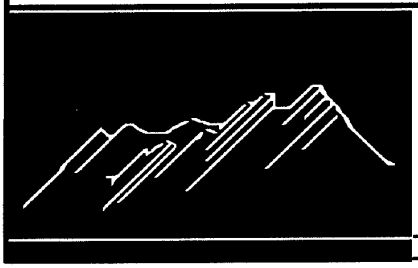
INSTITUTE OF COGNITIVE SCIENCE



Technical Report

University of Colorado, Boulder

A Field Study of Exploratory Learning Strategies

John Rieman

Institute of Cognitive Science
University of Colorado
Boulder, Colorado 80309-0344

Technical Report #94-07

8/94 draft: submitted to ACM Transactions on Computer-Human Interaction.
Author's address: Institute of Cognitive Science, Campus Box 344, University of Colorado, Boulder 80309-0344 (rieman@cs.colorado.edu).

A Field Study of Exploratory Learning Strategies

JOHN RIEMAN
University of Colorado

It has been suggested that interactive computer users find "exploratory learning" to be an effective and attractive strategy for learning a new system or investigating unknown features of familiar software. In exploratory learning, instead of working through precisely sequenced training materials, the user investigates a system on his or her own initiative, often in pursuit of a real or artificial task. The value of exploratory learning has been studied in controlled settings, with special attention to newly acquired systems, but there has been little investigation of its occurrence in natural situations or in support of ongoing learning. To address this question, a field study of the behavior and attitudes of computer users in everyday working situations was performed, using diaries and structured interviews that focused on learning events. The study showed that task-oriented exploration was a widely accepted method for learning, but that it often required support from manuals and from other users or system-support personnel. Exploration not related to a current or pending task was infrequent, and most users believed it to be inefficient.

Key Words and Phrases: exploratory learning, learning on demand, learning in the work place, diary studies.

1. EXPLORATORY LEARNING

In the early 1980s, when word processors brought the first wave of interactive computing into the office environment, it seemed obvious to many users and managers that these complicated, expensive machines could only be operated by employees who

This work has been supported by grant number IRI-9116640 from the National Science Foundation. A preliminary report of this study was presented at INTERCHI'93 conference, sponsored by ACM/SIGCHI under the aegis of IFIP in cooperation with IFIP TC.13, in Amsterdam, April 24-29, 1993 (Rieman, 1993).

had been formally trained in their operation. Learning to use the word processor in the office where the author worked required two days of classroom training, and employees who hadn't completed the course weren't supposed to use the machine. When a significantly revised version of the word processing software was installed, another day of off-site training was required. A change of hardware vendor brought yet another round of training. This was on systems with dedicated keyboards and textual menus, 8–inch diskettes holding less than 200K of data or program, and software that provided features numbering at best in the dozens: insert, overstrike, cut, paste, underline (with the daisy-wheel printer), simple mail merge, a rudimentary tabular facility, and a game hidden on a diskette by the hardware repair person during one of her many visits.

No one took classes to learn how to use the game, but it wasn't long before all the secretaries and anyone's visiting children knew how to play it. The repair person had shown someone how to boot the program, one or two fearless users played with it during coffee breaks until they had discovered most of its features, and the knowledge of how to use it was passed along within the work group.

Today it isn't uncommon for an office worker to use several major pieces of interactive software every day, including word processors, spreadsheets, electronic mail, graphics programs, and the windowing front end to the computer's operating system. We have calculated, roughly, that this may require a user to deal with a major software upgrade on the average of every six months (Franzke & Rieman, 1993). And each piece of software may have many hundreds, even thousands, of commands and options, the majority of which any given user might never use (Fischer, 1987; Nielsen, Mack, Bergendorff, & Grischkowsky, 1986). In the face of this overwhelming need to continually relearn job-critical software, do users still depend on formal classroom training to maintain their skills?

Anecdotally, the answer is obvious. Many users, perhaps most, now learn to use their productive software the way early users learned to use the hidden games: through trial and error, through interaction with other users, through occasional reference to manuals if they are available. In short, they rely on what can be broadly defined as "exploratory learning." The purpose of the research described in this paper is to put that anecdotal observation on firmer footing.

1.1 Studies of Exploratory Learning

The potential value of exploratory learning has been recognized since interactive computing first began to enter the work place. Malone (1982), Carroll (1982), and Shneiderman (1983) all noted the success of computer games and argued that other computer software should have some of the same characteristics. Mastering the software should be intrinsically motivating, features should be revealed incrementally, and the system should be at least minimally useful with no formal training.

Carroll and his colleagues began an extensive program of research in support of this goal in 1981. This work initially analyzed the learnability of office-system interfaces in situations where novice users were given no coaching or training (Carroll, Mack, Lewis, Grischkowsky, and Robertson, 1985). The research demonstrated that most users are willing and able to learn by exploration, although there are individual differences in exploratory aggressiveness (Carroll & Rosson, 1987; Carroll, 1990; see also Neal, 1987). It also showed that novices attempting to explore a totally new system often make major and unrecoverable errors, even with the aid of basic manuals and tutorials (Carroll and Mazur, 1986).

Carroll and his associates continued this line of investigation, focusing on a "minimalist approach" to training and manuals (Carroll, 1990). The approach has shown dramatic success in bringing new users up to speed in complex environments, and many users have expressed a preference for this kind of guided exploratory learning.

The details of exploratory learning of an interface in the absence of manuals or instruction have also been investigated. Shrager (1985; Shrager and Klahr, 1986) gave subjects the unstructured task of learning to operate a "BigTrak" toy, a six-wheeled truck with a built-in computer controller, programmed with a simple keypad. Shrager found a common pattern of behavior, beginning with orientation and continuing through the generation and testing of hypotheses about individual keys. Significantly, Shrager's undergraduate subjects were able to uncover the major functionality of the interface, but the task was time consuming and the behavior of more sophisticated controls was often not resolved.

Ongoing research by Lewis and Polson and their colleagues has also investigated uninstructed behavior where no manuals are available (Polson & Lewis, 1990). This

work has examined the actions of users faced with well-defined tasks and novel computer interfaces. In an early report on this research, Engelbeck (1986) suggested that the behavior of subjects faced with novel menus can be described by a label-following strategy, predicted by the identity heuristic in Lewis's (1988) EXPL theory. More recently, Franzke (1994) observed users who are novices to a particular software application, but who are competent users of the computing environment in which the software is provided. Franzke's work confirmed the power of the label-following strategy. She reported that users are quick to uncover basic functionality, but that their efforts to complete a task may be seriously blocked by misconceptions or poorly labeled controls.

1.2 Research Approach and Overview

Each of the studies described has taken the view that users can learn a system through exploration, and the more practically oriented research has investigated ways to support that learning through better software, manuals, or training. The research described in this paper approaches the issue from a different, complementary, direction. It asks: Within the work and computing environments currently available, when and how do users learn to use the software they rely on for their daily tasks?

In brief, the study was performed as follows. Diaries were maintained for one work week by fourteen interactive computer users, who represented a wide range of computing skill and job duties. The diaries used predefined categories covering computer interactions and all other work-related activities, with an explicit focus on learning events. There was no attempt to focus specifically on exploratory learning. Following the one-week diaries, a structured interview was performed with each user. The questions in the interview also focused on learning, but extended the scope of the study beyond the one-week time slice reported in the diary.

This approach distinguishes the current study from previous work on several dimensions. First, instead of observing novices' first experiences with systems, the study most often captured experienced users extending their knowledge of systems they used every day. Second, instead of performing a controlled comparison between types of manuals or training, or observing users forced to explore with no manuals at all, this study observed users who chose their learning strategies under the constraints of their

real work and the available learning resources. Finally, the study addressed a question frequently left unanswered by previous work: When users are blocked in their unsupported exploratory efforts, what do they do?

The results of the study have value in several areas. For software developers, the study points to areas where products could be strengthened for greater competitiveness. For managers and users, the study suggests training and organizational strategies that could improve the usability of current systems. Perhaps most importantly, for the purposes of ongoing research in our laboratory and others, the study has provided a broad description of exploratory behavior that helps to define more controlled laboratory experiments that accurately reflect the constraints of the work place (Franzke & Rieman, 1993; Franzke, 1994). As such, the study reflects an essential step in a research paradigm that progresses from anecdotal evidence, to focused field studies, to controlled laboratory studies, with a concomitant refinement of theory and hypothesis at each stage (Rieman, 1993, 1994).

2. THE DIARY STUDIES

The diary studies investigated users' exploratory and learning behavior in the course of one week of their ordinary work. In a diary study (Ericsson, Tesch-Römer, and Krampe, 1990; Rieman, 1993), informants record their activities on a standard log form, which breaks the day into fixed periods of time. Activities are allocated to predefined categories for later analysis. For this study, the daily log form was supplemented by additional forms on which the informants were asked to record learning events, if and when they occurred.

The "grain size" of the data exposed by the diary study is relatively large. Events that take several minutes are reported, but the low-level detail of those events is not clearly exposed. Thus, the field study gives evidence of the existence and context of learning events, but may not clearly expose their structure.

2.1 Method

3.2.1.1 Informants

Fourteen informants participated in the study, seven male and seven female. They are identified in this report as Informants 2 through 16, and are all referenced with

male pronouns, to help preserve anonymity. Informant 1 was a pilot subject whose data is not reported, and there was no Informant 13. The informants were selected to provide a broad range of experience and work demands, as shown in Table 1. All participants volunteered their time.

Table 1

Informants' Background and Experience Ratings

ID*	Exp†	Op. Sys.**	Computer Use	Position; Duties	Background
2	1	Unix/ Work-station	clerical	secretary; general	On-the-job and classroom training on Unix-based and dedicated word processing, spreadsheets.
3	2	Mac/ VMS	clerical	secretary; general	On-the-job training, some classes, on dedicated word processors, PC and Macintosh-based word-processors and spreadsheets.
4	5	Work-station /Unix	work/ research	Ph.D. student in computer science; AI research, writing	Industry experience in program development and support. Extensive use of minicomputers, personal computers, AI workstations.
5	5	Mac/ Unix	work/ research	Ph.D. student in computer science; AI research, writing	Industry experience in program development and support. Experience with Unix, PC's; learning Macintosh. Has taught computer science courses.
6	1	Mac/ PC	clerical	undergraduate in engineering, clerical assistant duties; home-work	Home experience with PC's. Work experience with word processing and spreadsheets on Macintosh. Some programming training.
7	5	Unix/ Work-station	work/ research	faculty member in computer science; AI research, teaching	Extensive academic experience with Unix systems and AI workstations. Limited PC/Macintosh background. No industry experience
8	4	Mac/ Unix	primary tool	research faculty in cognitive science; cognitive modelling, empirical research	Master's degree in computer science; has taught computer science courses. Macintosh user and Lisp programmer. Research focus on programmers, not software applications.

Table 1 (Cont.)

9	3	Mac/ PC	clerical	undergraduate in pharmacy, clerical assist.; homework, assigned duties	Work and home experience with several programs on PC and Macintosh. High-school programming courses. Aggressive learner of new programs
10	5	Unix/ Mac	work/ research	Ph.D. computer science researcher in industry; program development	Industry experience in program development and support. Extensive use of minicomputers, personal computers, AI workstations. Has taught computer science courses.
11	4	PC/ VMS	science support	faculty member in social sciences; research, teaching	Extensive experience with personal computers, both command line and graphical interfaces. Relies heavily on databases, word processors, and presentation software for teaching and research. Programs in database, system command language.
12	3	Mac/ VMS	science support	Ph.D. student in psychology; empirical & bibliographic research	On-the-job experience with minicomputer statistics packages, Macintosh spreadsheets, word processing, and graphics. Some programming training, occasional programming for research in statistics programs, HyperCard.
14	4	PC/ Mac	primary tool	financial analyst in industry; data analysis, predictions	Master's level training in information systems. Programming courses in FORTRAN, Pascal. Extensive experience with PC's and Macintosh, both as a user and support person. Frequent programmer in database and spreadsheet packages.
15	2	Mac/ VMS	science support	faculty member in psychology; empirical research, teaching	On-the-job experience with Macintosh word processing, graphics, and statistics. Some experience with similar programs on window-based PC's. E-mail on minicomputer. Has avoided learning to program.
16	4	PC/ Mac	primary tool	financial analyst in publishing; data analysis, work flow investigations, training for job sharing	Courses in programming (FORTRAN, COBOL, Pascal) and finance. Extensive experience evaluating Macintosh application software in a development environment. Heavy user of spreadsheets and databases. Involved in user support.

* ID 1 was a pilot subject whose data is not reported; there was no ID 13.
† Primary/secondary operating system; PC includes both MS-DOS and MS Windows.
** See Table 2 for explanation of experience ratings.

2.1.1.1 Computer use distribution of informants

Four informants, identified by the key word "clerical," in the computer-use column of the table, were selected as representative of the large population who uses word processors and spreadsheets in an office environment. Two of these users were professional secretaries in university academic departments, one using a Unix-based system, the other working with Macintosh equipment. Two were undergraduate student clerical assistants, doing secretarial and general office work as well as their class work. None of these users had extensive computer science training or skill, although the secretaries had professional level skills with the software they used in their work.

Three informants were selected to yield insights into the work of scientists who rely on computers for text processing, graphics, and some data analysis, activities that are in support of their main research interests. Two of these informants were faculty members in departments outside computer science (a social science and psychology), neither of whom was involved in computer modelling of cognition or other research that demanded extensive computer work. A third informant was a doctoral student in psychology, again in an area that did not demand a high level of computer skill.

Another three informants, whose computer use is described as "primary tool," represented the growing force of workers for whom the work environment is not simply improved but is entirely defined by the presence of computers. Two of the informants worked in business, where they did financial modelling and analysis that would have been impractical or even impossible before the advent of PC-based spreadsheets and databases, often in communication with mainframe accounting packages. These informants were both highly skilled users and programmers within the applications they used, but neither was a trained software developer. The third informant in this group was a cognitive science researcher, with a strong background in computer science, but with research activities in which computers were a tool for empirical analysis and cognitive modelling, not a topic of investigation in themselves.

Finally, four of the informants were highly experienced computer scientists, including a faculty member, two graduate students nearing completion of their Ph.D.s, and an industry researcher with a Ph.D. and several years software development

experience. These users perform work and research in the field of computers themselves. It was expected that the most extensive exploratory activities would be seen in this group.

2.1.1.2 Experience distribution of informants

The informants' experience with computers ranged from novice users who had worked with only one or two word processors to professional Ph.D. computer scientists with experience on systems as simple as a Macintosh and as complex as a Symbolics workstation. The years of experience for each user were difficult to determine, and the experience ratings defined more in terms of breadth of experience than depth. Table 1 lists the ratings for each user, and Table 2 provides a key to the meaning of the numerical ratings. The novice (narrow) category described the background of two of the users, who had only worked extensively with a few word processors, with some additional exposure to other programs such as spreadsheets and graphics. Two more users fell into the novice (broad) category, having more training and more extended experience with software in addition to word processing. The intermediate category, which also included two informants, was defined to be the lowest level that used programming (including operating system command files, data base queries, and other application-specific languages). Four informants fell into the advanced category, which required experience with a variety of systems and frequent work-related use of programming. Finally, the expert category included four professional computer scientists, all with Ph.D.-level training (two Ph.D.'s in progress) and all with industry experience in program development.

Table 2

Key to Experience Ratings

Rating	Number of Inf's	Description	Defining Experience
1	2	Novice (narrow)	Work experience limited to one or two word processors, e-mail, possibly a spreadsheet; no use of programming.
2	2	Novice (general)	Work experience with several types of software application, some formal training, no use of programming in job.
3	2	Intermediate	Work experience with many software applications and more than one system, programming training or experience, programs infrequently.
4	4	Advanced	Work experience with many software applications and systems, frequent use of programming in work (including traditional languages or application languages such as spreadsheet macros, complex database queries, system command files), has supported other users professionally.
5	4	Expert	Professional programmer, familiar with many systems, extensive formal training in computer science, has taught or supported other users.

2.1.1.3 Operating system distribution of informants

In addition to having a wide range experience, the informants also used a variety of computer operating systems as their primary system. Systems represented included Macintosh (7 users), PC's (with or without MS Windows; 3 users), Unix or VMS (3 users), and high-end scientific workstations (i.e., Sun, Symbolics; 1 user). Systems on which users had secondary access or significant past experience, typically for e-mail purposes or on a home system, included Macintosh (3 users), PC's (2 users), Unix or VMS (7 users), and high-end scientific workstations (2 users). For some users it was difficult to say which was the "primary" system; in these cases the system listed is the one in which the user reported the most computing during the log week.

2.1.2 Materials

Informants maintained their daily log of activities on a log sheet of the form shown in Figure 1. The log was on 11-inch by 14-inch paper, and the hours on the left side corresponded to the informant's typical working hours, with a few hours

Day: *TUES - 9/15*

I.D.: *7*

Activity Log:
Fill in Every Half Hour

Categories: Fill in at End of Day

		Talk, in Person	Talk, Phone	Meetings	File, Organize	Fill in Forms	Copying	Paper Mail	E-Mail	Word Process	Spreadsheet	Other Computeing	Breaks/Personal	Reading	Class	Other
8-8:30	*GOT COFFEE* *CHECKED E-MAIL*								*				*			
8:30-9	*PHONED ABOUT CAR* *MORE E-MAIL*		*						*							
9-9:30	*MET WITH STUDENT* --			* * *												
9:30-10	*AI CLASS* --														* * *	

Figure 1. The beginning of a diary log sheet for one day. The informant records activities, and the investigator assigns categories during the end-of-day debriefing.

"Eureka" Report

For Computers, Phones, Copiers, Fax Machines, Staplers, Clocks, Thermostats, Window Locks, Cameras, Recorders, Adjustable Chairs, and other Strange Devices

I.D. ____7____. Date & Time: *9/15 at 11 a.m.*

Describe the problem you solved, or the new feature you discovered, or what you figured out how to do:

GOT COPIER TO PUT STAPLE IN RIGHT CORNER!!!

How did you figure it out? (Check one or more, explain)

___ Read the paper manual
___ Used on-line "help" or "Man"
_X_ Tried different things until it worked
___ Stumbled onto it by accident
___ Asked someone (in person or by phone)
___ Sent e-mail or posted news request for help
___ Noticed someone else doing it
___ Other

Explain:

CAN'T DECODE "INTERNATIONAL" COPIER SYMBOLS.

Figure 2. A "Eureka" report slip, for noting successful or attempted learning events.

extra. Usually this was from 7 a.m. to 7 p.m., but some subjects worked a later schedule and used a sheet that ran from 10 a.m. to 10 p.m. In addition to the log sheet, informants were supplied with a stack of blank "Eureka Slips," as shown in Figure 2. These were printed on various bright colors of paper and held together with a colorful spring clip. The colors were intended to make the slips more visible and increase the likelihood that informants would remember to fill them in when appropriate.

The materials were designed to focus on learning, without specifically narrowing the issue to exploratory learning. There was no specific category on the log sheet in which to record time spent exploring computer systems. The sheet included an "other" category as well as categories for common systems (word processing, spreadsheets, etc.), and these were intended to cover all computer activity. The Eureka slips narrowed the focus somewhat, being clearly concerned with learning, but they were broad enough to cover many kinds of learning activity, including both task-oriented and task-free exploration. The categories on the log and on the Eureka slip were initially selected based on anecdotal evidence of learning strategies, and were refined after a small pilot study.

2.1.3 Procedure

The investigator gave a brief description of the log materials to the informants at the time they were recruited. A log period was scheduled at that time. As much as possible, the log period was scheduled to be a week of "ordinary" activity for the user. A week when an informant would be out of town for a conference, for example, would not be logged (unless travelling to conferences was a common activity for that user); but there was no effort made to prefer or avoid times when the user had just started work with a new piece of software.

Immediately before the log period began, the investigator met with the informant and explained how the logs were to be used. Informants were to fill in the left-hand side of the log sheet as each day progressed, using their own terminology for the work they were doing. To protect their privacy, they were allowed to mark any period as "breaks/personal" and provide no further information. Informants were instructed to fill in a Eureka slip whenever they learned something new about their computer system

or some other equipment in their work. They filled in the slips completely, including the description of the problem, the strategy used to solve it, and any additional comments. Informants were also asked to use Eureka slips to record failed attempts to solve problems.

At the end of each log day, the investigator met with the informant (in person for all but two informants, and over the phone for those two (10, 14)). The investigator spent 10 to 20 minutes talking over the day's activities and assigning them to the categories on the right-hand side of the log sheet. The day's Eureka slips, if any, were also discussed briefly during this meeting, and the investigator often made additional notes on the back of the slips, or occasionally corrected a strategy assignment. If the discussion of logged activities revealed a learning episode that had not been recorded on a Eureka slip, the investigator and the informant would fill one in for that episode.

2.2 Results: Logged Time

This section presents summaries and analysis of the log data. The data presented include only logged time not marked as breaks/personal.

All but two of fourteen informants kept their logs over a consecutive 5-day period. One informant (3) kept a log over 4 1/2 work days plus 1/2 work day; for another (10), one day of the five was eliminated from the data because the log was only partially and inaccurately completed. For most informants, the log skipped weekends and holidays, although some informants worked over the weekend and logged that period.

2.2.1 Overview

An overview of the logged time is given in Table 3. The mean number of hours logged, after personal time was subtracted, was 34, ranging from 27 to 39.5 hours per informant. Within those 34 hours, informants spent an average of 17.3 hours, or 50.9 percent of their time, working with computers. For the purposes of this analysis, computers were defined narrowly to include PC's, minicomputers, and mainframes. They did not include phone systems, VCR's, fax machines, copiers, and similar hardware that typically has a dedicated internal microprocessor and a multi-function interface. The format of the log sheet made it difficult to accurately determine time

spent in these activities, since individual sessions with these machines typically took much less than the log's 30-minute basic interval. Computing time for individual informants ranged from 1 hour (Informant 6) to 34.4 hours (Informant 5). As a percentage of their total log time, computing time for individual informants ranged from 2.5 (Informant 6, 1 hour out of 28.5) to 91.7 percent (Informant 8, 27.75 hours out of 30.25).

Table 3

Hours Logged During the Diary Study

	Total	Mean	StdDev	Min	Max
Hours Logged, Less Personal Time	476.5	34.0	4.27	27.0	39.5
Hours of Computing	242.6	17.3	8.41	1.0	34.4
% of Log Hrs Spent Computing	50.9	50.9	24.3	3.5	91.7

2.2.2 *Time in Application Categories*

As part of the log, informants recorded the category of computer application they were working with. That data is summarized in Table 4. The categories should be self-evident except for "special applications," which included mostly minicomputer-based software that was custom programmed for some business function. Operating system activities included disk formatting, directory management, and backups; working with batch or command files was categorized as programming. The news and on-line information category reflected mostly internet news reading, but one subject also spent time looking at the University of Colorado's on-line general information.

Table 4

Hours Logged in Software Categories During the Diary Study

Software Category	Total Logged			Percent of Informants' Computing Times			
	Hours	% Ttl	Inf's*	Mean	StdDev	Min	Max
Word Processing	87.5	36.1	12	37.7	30.6	0	88.5
Programming	28.5	11.7	4	7.6	25.9	0	97.3
E-Mail	26.0	10.7	9	15.3	22.1	0	60.2
Database	25.8	10.6	4	7.6	14.2	0	40.6
Special Applications	24.0	9.9	4	13.6	26.3	0	85.9
Spreadsheets	17.3	7.1	5	5.9	14.7	0	55.2
Graphics	11.5	4.7	4	4.0	9.4	0	33.9
Telecom (up/dnload)	9.0	3.7	4	2.4	5.0	0	16.7
Operating System	6.8	2.8	7	3.0	5.7	0	21.0
Games	3.5	1.4	1	1.4	5.4	0	20.3
News/On-Line Info.	2.5	1.0	3	1.2	2.5	0	8.7
Unknown	0.3	0.1	1	0.1	0.4	0	1.7
total	242.6		14				

*Number of informants who logged time in the category.

The left side of the table is a simple total of all times recorded, with a count of informants who reported spending time with each category. This is an indication of the study's application sample characteristics. The category accounting for the most hours is word processing, which makes up 36.1 percent of the total computing hours logged. No other single category is a standout, although there is a range of 10 to 1 between the group of four categories that each make up about 10 percent of the remaining time (programming, e-mail, databases, and special applications) to the least popular application (network news and on-line information at 1 percent).

The right side of the table was developed by determining the percentage of each informant's total computing time that was spent in each category, then calculating the averages of those percentages. This is a rough indication of the sample population's work habits, and its most notable characteristic is its variance. The mean percentage for word processing, where informants on the average spent the largest proportion of their computing time, is 37.7 percent, but with a very great range. Two of the fourteen informants did no word processing at all. Of the remaining categories in Table 4, only

electronic mail shows a mean percentage (15.3) and a number of informants (9) that even approaches a consensus.

2.2.3 Time Spent in Task-Free Exploration of Computer Systems

A central question addressed by the study was whether the informants found their computing environments intrinsically interesting enough to motivate exploration in the absence of any work-related tasks. To avoid biasing the informants toward this kind of behavior, the log sheets did not include an express category for "exploration." However, any extended exploration would have been noted during the investigator's daily debriefings, and it would have generated a number of Eureka slips, described in greater detail in the next section. These data sources show no evidence of time spent by any informant in task-free exploratory learning of an interface. Informant 12 spent 15 minutes browsing the university's on-line information system with no set goal, but this was an exploration of the information, not of the interface. Informant 11, who had recently acquired new software, spent more than 5 hours doing task-oriented exploration of the software's capabilities, out of 15.5 hours total computing time. The tasks were small and artificial, and the informant admitted during the daily debriefings that his work-related need for the software really wasn't great enough to justify the time he was spending with it. Three other informants (5,10,16) were working with new software during the log week, but all of them structured their activities entirely around real, current tasks.

2.3 Results: the Eureka Slips

A total of 78 learning events were recorded on Eureka slips. Of these, 18 involved copiers, phone systems, or other equipment whose operation was not included in the calculation of each informant's "computing time." Because no background information was collected concerning these systems and users' related experience, those 18 Eurekas are not included in the detailed analysis that follows.

Table 5

Hours and Eurekas Logged During the Diary Study

	Total	Mean	StdDev	Min	Max
Hours of Computing	242.6	17.3	8.41	1.0	34.4
Number of Eurekas	60	4.29	3.95	0	15
Eurekas per 8-Hr Computing (E/8hr)	1.73	1.73	1.28	0.00	4.13

The distribution of the 60 computer-related Eurekas across informants is summarized in Table 5. One informant (5) reported 15 Eurekas; the counts for the other informants ranged from 0 (Informants 6 and 15, who both reported non-computer Eurekas) to 8 (Informant 11). As the table shows, the time spent in computing ranged from 1 hour to 34.4 hours. This information can be used to normalize the Eureka count for each informant, providing an indicator of learning events per computing time. The measure "Eurekas per 8 hours of computing" (E/8hr) was chosen as a meaningful unit. For this measure, the range is 0 to 4.13.

The E/8hr scores for three users stand out: Informants 5 (with a raw Eureka count of 15), 11, and 16 have scores of 3.48, 4.13, and 3.43, respectively. The E/8hr scores for all other informants are less than or equal to 2.30, with a mean of 1.19 (StdDev=0.8). This suggests that there may be two distinct types of users, or perhaps two distinct situations in which users find themselves. In either case, by far the most prevalent case is that of the user who learns new things about a system infrequently, perhaps only two or three times a week for a user who spends as much as 50 percent of his or her time working with a computer.

Table 6

Eurekas per 8-hour of Computing, by Informant Categories

	Mean	Mean*	Values
Overall	1.73	1.19	
by Gender			
female	1.98	1.26	
male	1.48	1.14	
by Experience			
1 (novice)	.905	.905	(1.81, 0)
2	1.15	1.15	(2.3, 0)
3	2.31	1.41	(.62, 4.13, 2.19)
4	2.08	1.42	(1.73, 1.1, 3.43)
5 (expert)	1.72	1.14	(1.59, 3.48, .89, .93)
by Computer Use			
clerical	1.18	1.18	(1.81, 2.30, 0, .62)
science support	2.10	1.11	(4.13, 2.19, 0)
primary tool	2.08	1.42	(1.73, 1.1, 3.43)
work/research	1.72	1.14	(1.59, 3.48, .89, .93)
by Primary Operating System†			
Macintosh	1.47	1.14	(2.3, 3.48, 1.73, .62, 2.19, 0, 0)
MS-DOS/MS Windows	2.89	1.10	(4.13, 1.10, 3.43)
Unix/VMS	1.21	1.21	(1.81, .89, .93)
Workstation	1.59	1.59	(1.59)

* mean of all values except the three scores exceeding 3.0 E/8hr.
† not necessarily the system in which all Eurekas occurred.

In Table 6, the E/8hr scores of informants are broken down by gender, experience, type of work, and principal operating system. There are no notably large differences between the scores within any category, especially after the values for the three unusually high E/8hr scores are removed from the means.

The weak trend toward a greater number of learning events per 8-hours of computing by more experienced subjects is strengthened when the relative complexities of the Eurekas are considered. Novice users recorded many simple Eurekas, such as learning how to select several files at one time on a Macintosh. Experienced users sometimes recorded complex Eurekas, such as getting a simple "hello-world" program to run in a new data-base language. It seems likely that the complex Eureka involved learning many more individual facts than the simple one, although both were reported on single Eureka slips.

Table 7

Distribution of Eurekas by Strategy
(See Figure 2 for full text of strategies)

		Strategies checked							
	Total	Try	Read	Ask	Help	Stmbl	Notice	E-mail	Other†
Totals									
all informants	60	37	26	16	9	4	3	2	7
all but E/8hr > 3.0	30	15	11	8	3	2	3	0	2

* total number of single Eureka slips in the category.
† several user-defined strategies, no more than 2 Eurekas using any one strategy.

Table 8

Combinations of Strategies Shown on Eureka Reports

		In combination with... *							
	Alone	Try	Read	Ask	Help	Stmbl	Notice	E-mail	Other†
Try	9		17	7	8	2	1	2	5
Read	8	17		2	5				2
Ask	9	7	2		2	2		1	
Help		8	5	2			1	1	1
Stmbl	1	2		2					
Notice	2	1			1				
E-Mail		2		1	1				1
Other	1	5	2		1			1	

* 10 Eurekas reported more than two strategies; for these, all combinations of the strategies were incremented; i.e., try-read-ask added 1 to try-ask, 1 to try-read, and 1 to read-ask.

Table 7 presents what may be the most interesting breakdown of the Eureka data, showing the counts of how many times each strategy was used to solve a problem. Many of the Eureka reports showed that more than one strategy had been applied, so the count of strategies exceeds the total number of Eurekas. Three strategies dominate the reports: trying things out, reading the manual, and asking for help. Using on-line help falls into a middle ground, and it is more common for the three aggressive explorers than for the average informant. None of the other strategies appear to be generally useful.

Table 8 shows the counts of strategies used alone and used in combination with other methods. The most striking value in this tabulation is the number of times that trying things out is combined with reading the manual. Seventeen out of the total of 60 Eurekas, more than 25 percent, reported this approach. Trying things out is also the

only method that is found in combination with every one of the other strategies. Conversely, the "Alone" column shows that trying things out, reading the manual, and asking for help, identified in Table 7 as the most common strategies, are also the only strategies with a significant likelihood of resolving a problem without recourse to some other method. It is especially interesting that on-line help is never successful alone.

In Table 9, the Eureka strategies are categorized according to the gender of the informants. There is some suggestion from the tabulation that female users are more likely to ask for help, while males are more likely to read the manual. But this data is skewed by the fact that one of the high-E/8hr informants was a female who asked for help 6 times while working with a minicomputer system that had no manual, while another high-E/8hr user was a male who used the manual for 9 Eurekas, many of them while learning a system that no one else in his office was using.

Table 9

Distribution of Eurekas, by Strategy and Gender

		%							
	Total	Try	Read	Ask	Help	Stmbl	Notice	E-mail	Other†
female	28	17	8	11	6	0	2	2	4
male	32	20	18	5	3	4	1	0	3

2.4 Discussion

The data described here are largely self-reported, and hence need to be regarded with some caution. Although the informants were generally cooperative, they clearly had somewhat different ideas of what constituted a "Eureka." The daily debriefings helped to regularize this view (this effect was weakest with the two informants who were debriefed over the phone), but there were also other factors that could have affected the number of Eurekas recorded. The informant who figured out how to get a new database program to run probably wouldn't have had time to fill in a separate Eureka report for each fact learned. Similarly, an informant under pressure of an

impending deadline might learn something and forget to record it at the time or mention it in the daily debriefing.

The weakest result, then, is the rate at which users typically discover new things about their system. The data suggests that this happens infrequently, perhaps averaging only 1 learning event for every eight hours of computing time for the typical user. Aggressive, experienced users may reach rates as high as 3 or 4 events per eight hours, and we can multiply this by some unknown factor that reflects the greater complexity of the things learned.

The data are stronger when viewed as a sampling of learning events that did occur, without regard to whether all such events were recorded. In this light they are noteworthy for the similarity of learning strategies recorded across a very wide range of situations and users. There is significant evidence that the three preferred strategies are trying things out, reading the manual, and asking for help. Conversely, users are quite unlikely to learn things by noticing other users' activities, or by stumbling onto things in the interface, or by communicating via e-mail or network news. On-line help falls into a middle ground.

Going beyond the cautionary remarks concerning self-reporting, the data are limited in that they represent only one week of observation for each of the informants. As described earlier in the section, an effort was made to choose an "average" week. However, to supplement this data it will be useful to consider the informants' learning strategies over a longer term. This is the topic of the next section.

3. INTERVIEWS

The diary logs reported a thin time slice of the informants' working life, covering only five days. To extend the investigation of each informant's behavior beyond that short period, another research technique was chosen: the structured interview. The questions of the interview covered essentially the same topics that had been the focus of the log period, but with some extensions to exploratory behavior outside the work place.

3.1 Method

The investigator interviewed each of the fourteen informants described in the preceding section at a time convenient to them, after they had completed their log.

Typically, the interview took place within two or three days after the log period. The interview was structured around a series of prepared questions (Table 10). The investigator would follow up informants' initial answers on each question by encouraging him to give examples or reasons. When necessary, the interviewer would clarify a question by giving examples of the kinds of answer that might be appropriate.

All but one of the interviews were taped and transcribed by the investigator. They ranged in length from 20 minutes to 90 minutes, and the transcriptions ranged from 1500 to 7500 words, for a total of 66,000 words. The transcriptions and notes taken during the interview were examined to produce the summaries in this section. (The answers to the first question, on background, were described in Section 2.)

Table 10

Questions Asked in the Structured Interview
(The wording of each question varied slightly from one informant to another.)

1. Can you give me some background information about your experience with computers?
2. When you get a new piece of software, how do you learn to use it?
3. When you're using a program that you already know, and you run across something that you need to do but don't know how, how do you figure out how to do it?
4. Do you ever play around with computers, just try things out to see what they will do?
5. On the Eureka slips there is a checklist of ways you might use to learn something new. Can you give me specific examples of times you've learned things using each of those strategies?
6. Do you ever "explore" things that aren't related to computers? For example, do you go shopping when you have nothing to buy, or wander around with no fixed itinerary when travelling, or take apart things just to see how they work?
7. I'm especially interested in whether people "explore" computer systems, that is, do they try to find out what programs do even if they don't have any need for the knowledge. Do you have any further thoughts on that?

3.2 Question: Learning New Software.

After the informants described their background, the interviewer asked them, "When you get a new piece of software, how do you learn to use it?"

3.2.1 Four approaches to learning

The informants identified four main ways to initially familiarize themselves with a new piece of software:

- reading the manual, usually explicitly identified as being done in conjunction with one of the other methods (8 users:3,4,5,8,9,10,11,12)
- exploring its functionality, usually in the context of actual tasks (7 users: 4,5,6,7,10,11,14),
- working through the supplied tutorial materials (6 users: 5,6,11,12,14,16),
- having an experienced user demonstrate the package (5 users: 2,3,8,9,15),
- learning the basics initially, then learning more as the task demands (5 users: 9,10,11,12,16).

In addition to the five main strategies, a few users (2,3,9,14) had taken classes on some occasion, but this was always an exception to the way they expected to learn a program. One user (12) said that watching other users was a way to learn when no training materials or documentation were available.

As informants recalled examples of past learning activities, it became clear that the lines between the approaches were not clearly drawn. Availability of training materials or experienced personnel partly defined the strategies selected. Where tutorials were available, users had sometimes worked through examples as designed, other times they had used the tutorials and examples as a foundation for task-free exploration, while in still other situations they had used them as guide to a package's functionality, then moved on to real-world tasks.

3.2.2 Novice versus expert approaches

Several of the less experienced users (6,9,12) described their approach to learning a new piece of software in terms of a few past instances, which had offered different learning options. Informant 6, for example, stated that he would usually "fool around with a new program until it works," and he recalled a recent example where he didn't have a manual. But he also remembered doing a tutorial on another occasion. Informant 12 remembered learning one package by watching other users in a common computing environment, where no manual was available. The same informant noted that learning

was much faster for another package, where he had been able to use a tutorial and the manuals. In general, these users did not seem to have developed a consistent approach to learning a new package, or at least none they were aware of.

The more experienced users, on the other hand, had clearly defined learning strategies, which they stated without hesitation. The informants who had worked primarily with PC or Macintosh-based software typically had a single approach to using the mixture of documentation that has become fairly standard in that arena. Informant 11 always starts with the video, if there is one, then follows the installation instructions and does the tutorial, then tries small sample tasks, then turns to larger, real tasks. Informant 16 follows the installation instructions (the "getting started" guide) and looks through the on-line tutorial, if there is one. He then begins to use the program, with the manual and detailed tutorials as fall backs for tasks he has problems with.

Informants who had worked with a wider variety of systems, including Unix and other time-shared computing environments, selected between two distinct strategies depending on the characteristics of the package they were learning. Informant 5 specifically identified two kinds of software: totally novel packages ("out of the blue"), for which the tutorial and examples were required to reveal the software's function, and packages that were similar to known systems, such as editors, for which task-oriented exploration was the preferred learning approach. He further identified two "modes" of learning behavior: *project mode*, where getting the current project completed was paramount, and *tool mode*, where efforts were directed at learning the new tool, typically by redoing a task that had already been completed with another tool. "Before I sit down at the terminal I know which mode I'm in," he said.

For informant 10, only simple software was worth a quick exploratory foray, while large, complex packages demanded an in-depth understanding of the manual and the program's philosophy. He felt large systems were not worth investigating without a real-world task to guide and justify the learning effort. For these packages, he identified a sequence of investigatory activities, beginning with the manual, proceeding to breadth-first, task-free investigation of menus and controls, and finally moving into depth-first, task-oriented work. Informant 4 made a similar distinction between simple and complex packages, stating that reading and understanding, including an understanding

of the source code, was preferred to exploration for major software applications where the code was available.

3.2.3 Task-oriented versus task-free learning

Of the seven users who identified exploration as one method of learning a package, six explained (some in response to the interviewer's follow-up question) that they performed this exploration in the context of tasks. The seventh (14) did not clearly respond to the follow-up question. One user (10) performed both task-oriented and task-free exploration, as described above.

For most users, the most effective approach was to use their own tasks. Informant 5 used his own tasks because in that "demand-driven" mode he would not waste time learning what wasn't needed. Informant 11 would sometimes begin exploration with simple, trial tasks, but would soon progress to larger, real tasks. Informant 16 would look at example tasks provided in the tutorials because they demonstrated the software's functionality, but he postponed learning until a real need arose. "I just keep them in mind if I want to do something like that," he explained, "because I know the tutorial would walk you through those examples," Only Informant 4 claimed a preference for the sample tasks provided with the system, reasoning that the data for the examples had already been entered.

3.2.4 Making use of manuals

Although eight informants mentioned using manuals to learn new software, only three (4,5,10) described situations in which they would start by reading the manual in depth. These users were all highly experienced computer scientists, and the occasions when they would read the manual typically involved learning a complex new programming environment (although Informant 10 would prefer to read the manual for in-depth understanding before tackling any large-scale application).

For the other informants, the manual was a supplement or a support to other strategies. Informant 11 would scan through the manual, but only after exploring the program for a while: "I take the manual someplace away from the computer... just to browse through it and look for features that might be interesting or useful later on."

Informant 8 liked to get an overview of the program through the manual, then have an experienced user demonstrate how to get the package running.

For the majority of users, the manual was most valuable as a fall back when they ran into problems, the situated investigated in Questions 2 and 4. Informant 16 expressed the consensus opinion of the value of manuals for initially learning about a program: "Reading the User's Guide is, to me, useless. I only use the User's Guide when I want to do something specific."

3.2.5 Time constraints

Many of the users explicitly identified time as a constraint on their learning activities. Informant 2 didn't like "reading through all that stuff [the manuals] and trying to find the answer to my problems." He preferred a personal demonstration. As noted above, Informant 16 never read the manual until he had a specific problem to resolve, and he initially looked at the examples only to see what could be done. However, he liked the on-line tutorials, "because they're short, and they show you a lot of things right up front."

Informant 3 volunteered his opinion of unstructured exploration: "It's not that I don't like to do that, but I feel like if I'm working, it's not a good use of time." Informant 7 described an extended exploratory session driven in part by curiosity but also guided by a "cost/benefit analysis." His initial impression of the software was negative because of obvious problems, but he decided to explore further to see if it contained features that might make the earlier problems worth overcoming.

Informant 5, who had mentioned the efficiency of the "demand-driven" approach to exploration, also described how he learned systems "incrementally": He would discover a problem, try to solve unsuccessfully to solve it for a while, then continue with his work. When the same problem came up again on another day, he would take a few minutes to try a different solution, but would give up again if that failed. "You know, I want to get this thing edited," he said, referring to the word processing package he was currently learning "and if I can't find a feature fast, I'm just going to forget it."

3.2.6 Summary

When learning a new piece of software, inexperienced users are likely to select whatever method is available to them. More experienced users, however, select the learning strategy that they believe will help them acquire the skills they need with the least possible investment of time. With relatively simple or standard packages and experienced users, exploration in the context of the user's immediate task may be an effective first strategy. With complex, novel packages, however, experienced users prepare themselves for task-oriented exploration by reading the manual, working through on-line tutorials where available, and availing themselves of their colleagues' expertise through brief demonstrations.

3.3 Question: Resolving Problems.

For many of the informants, there was no clear distinction between learning a new piece of software and resolving problems with a software package they had been using. This was the topic of the next question: "If you're working with a program that you already know how to use, and you run up against something that you don't know how to do, how do you figure out how to do it?"

The three strategies observed most often in the logged data also dominated the answers to this question:

- trying things out (exploration) was clearly identified as a first strategy for 9 users (3,4,5,6,11,12,14,15,16). An additional 2 users (3,8) described their first action as a combination of looking in the manual and trying things out.
- looking in the printed manual was a first strategy for 3 users (2,9,10) and a second strategy for 7 (2,4,6,11,12,14,16).
- asking for help was a first strategy for 2 users (3,7), a second strategy for 5 (2,6,8,9,15), and the third strategy for 4 (4,10,11,14).

Additional strategies identified were working around the problem (informants 4, 5, 10, and 12, the first three of whom would sometimes use programming for the work-around -- although informant 4 also praised the efficacy of white-out), using on-line help (informants 5, 7, and 11), and looking at the source code (a first strategy in complex

systems for informant 4, and a second strategy, after asking colleagues, for informant 11).

Note that some informants identified alternate "first" and "second" strategies, depending on the problem, time and resource availability, and in two cases (2,3), their mood at the time the problem arose. Informant 2: "It depends on my mood [laughs], if I want to talk to somebody or if I want to read it."

Many informants distinguished between things they would immediately ask system support personnel to handle and things they would try to handle on their own. Hardware problems were almost always referred to systems support. "That's sort of their job," explained informant 12.

3.4 Question: Task-Free Exploration of Computer Systems

All but one of the informants answered that they did little or no exploration of computer systems except for the purpose of performing current or impending tasks. The one informant (11) who identified this as a common activity had, according to the diary log, spent one third of his computer time exploring new software. Other examples he gave of exploratory behavior were changing the appearance of his on-screen work space, "about once a day," and producing graphs or charts for course materials that were unnecessarily decorated. "I really don't leather-look bar charts," he admitted. "And I don't really need a sophisticated, vector-based graphics program." He specifically stated that he was a little embarrassed about this behavior, because he knew it wasn't productive.

One other informant (9), an undergraduate, said he liked to do task-free exploration, but didn't have much opportunity. "Almost every program, I try to find something fun about it," he explained, giving the example of learning to use the sound-recording capability of his department's Macintosh to record alarm messages in his own voice and the voices of his co-workers.

The remainder of the informants answered either that they never or almost never did that kind of thing, at least not in their current situation. "Only once have I ever done that in my life," said Informant 5, and then gave the details of his experience with a Mandelbrot set program. Four of the informants (4,8,10,14) recalled doing more task-free exploration in the past.

Essentially there were two reasons expressed for not exploring computer systems. The most common reason was time constraints. Seven of the informants (3,4,6,7,8,14,15) specifically said they didn't have the time or they felt that exploration was not productive, at least not in their current job.

The second clearly expressed reason stated for avoiding task-free behavior was that it was simply not interesting. "I use [computers] only as an instrument, and I don't see them as something fun," explained Informant 15. A similar attitude was expressed by Informant 8: "I don't explore the actual computer all that much. Of course, the computer's a tool." And Informant 12, while admitting to have tried a few adventure games, ranked the computer low on the scale of interesting activities: "For leisure I like to run around outside or watch TV. I wouldn't normally come up with the idea of switching on the computer to entertain myself."

3.5 Question: Eureka Strategies

The informants had stated their preferred strategies for resolving problems, and these preferences were largely validated by the Eureka counts. It will be useful to consider the the reasoning behind these preferences. Were the preferred strategies simply the only ones the informants had ever tried? Or, more likely with the experienced users, had they tried many strategies but found the preferred ones most efficient?

To investigate these questions, and to provide the informants with a different index into their exploratory experience, the interviewer asked: "You've been using these Eureka slips in the diary study. Can you remember specific occasions in the past when you learned something through each of the methods listed on the slip?" For strategies discussed with users earlier in the interview, the answers were usually short, but for other strategies the question provided a springboard into discussion of the strengths and weaknesses of the approach.

The informants' answers indicated that most of them had tried the strategies available to them, but the ease with which they recalled specific instances echoed the strategic preferences they had stated in Question 1. (In the interviews, this question was asked after Question 4, Task-Free Exploration, so informants were not so likely to

immediately recall what they had just said in Question 1.) The results of the question are summarized in Table 11 and discussed in detail below.

Table 11

Answers to Question 4, "Can you recall examples of learning things using each of the categories in the Eureka report?"

Strategy	Used		Not Used	
Read paper manual	all			
Use on-line help or man	*yes, praise or no comment* 2,5,9,14	*yes, but problems* 4,7,8,10,11,12,16	*tried, doesn't work* 3,15	*no, no comment* 6
Tried things until it worked	*yes, de novo* all but three	*yes, with manual* 8,9,10		
Stumbled onto by accident	*yes, in interface* 9,11,12	*yes, in manual* 4,5,8,15	*maybe, can't recall* 2,3,6,7,10,14,16	
Asked in person or by phone	*colleagues or sys. support* all			
Sent e-mail	*unqualified yes* 5	*yes w/ reservations* 4,8,11,12	*no, too slow* 2,3,12	*no, no comment* 6,7,9,14,15
Posted to network News			*no, for reasons* 2,3,4,7	*no, no comment* 6,8,10,12,15*
Noticed someone else	*yes, serendipitous* 2,4,5,7,9,12,14,15,16		*only in training or demo* 6,8	*no, no comment or can't recall* 3,11
Other	*yes (usr grp, video, class)* 4,11,16			*no, can't think of any* 6,9,15**

* Informants 5,9,11,16 didn't answer the "net news" question

** Informants 2,3,5,7,8,10,11,14 didn't answer the "other" question.

3.5.1 Strategy 1: Read the paper manual

All informants recalled instances of solving problems by reading the software's manual. Only one user, who had stated a strong preference for exploration and asking for help, hesitated briefly before answering affirmatively and providing the example. For many users, the most useful manual was a commercial "how-to-use-Program-X" sort

of manual, written by someone other than the software manufacturer. In addition, users of networked systems often maintained written notebooks of procedures learned in training courses or from interactions with system support personnel.

3.5.2 Strategy 2: Used on-line "Help" or "Man"

All but one of the informants had tried on-line help or man, but for the majority of them it was not highly regarded. Three users gave an example of usage without any qualification (Informant 2 in Unix, 9 in Word Perfect, and 14 in DOS), and one experienced user (Informant 5, user of DOS, Macintosh, and Unix) described on-line help as his first fall back after trying things failed. The remaining comments were strongly negative or qualified.

Two users, both heavy Macintosh users, had tried on-line help and decided it was useless. "It never, ever, ever works for me," stated Informant 15, explaining that he either didn't understand the help message or else it clearly didn't answer his question. Informant 3 also complained that the messages were difficult to understand, and that they gave too much useless information: "I mean, you ask it for help and it tells you everything in the universe."

The remaining users recalled instances of finding things using on-line help, but qualified their opinion of the approach. Informant 11 would use on-line help on DOS systems, but only if the manual wasn't readily available. Five informants would use on-line help for some systems (VMS, Unix, Macintosh, SAS), but not for others (Unix, Macintosh, DOS).

Comments on Unix man, even by informants who used it, were negative (except for Informant 2, who had no positive or negative comments): Informant 4 uses man but finds it "generally useless." Informant 7 said it "doesn't usually amount to much." Informant 8 would "rarely use the man stuff" and gave an example of how it failed to resolve a problem. Informant 10 thought man was "the worst thing in the world."

Informants 4, 8, and 12 would use on-line help for command-oriented systems (Unix, VMS, SAS) but not for the software they had used on the Macintosh. Informant 4 thought the Mac help was "crap," especially the balloon help: "I haven't found any of those messages useful." Informant 8 would rather use the paper manuals, while Informant 12 preferred to explore and complained that Microsoft Word's help kept

popping up when it wasn't wanted. On the other hand, Informants 10 and 16 would use on-line help with the Macintosh but not with command-line systems (Unix and DOS, respectively).

It's interesting to note that the three informants (5,10,16) who liked Macintosh on-line help are users with extensive experience, including strong backgrounds in command-line systems (Unix and DOS). On the other hand, three of the five informants who did not like Macintosh on-line help (3,12,15) had limited experience with command-line systems. Informants 8 and 4 were exceptions to this pattern, having strong command-line experience but disliking Macintosh help.

3.5.3 Strategy 3: Trial and error

All informants recalled instances of trying different things until they had resolved a problem. Many of the examples had been given in answer to an earlier question or had arisen during the diary study, and there wasn't a lot of further discussion. Three of the informants (8,9,14) noted that they often used trial and error to disambiguate information from the manual. (Informants 8 and 3 had made the same comment in response to question 1).

One of the highly experienced informants (10) noted that this approach was common in programming (it's one definition of "hacking"). He made the distinction between a "quick and dirty" approach and doing a job well. He associated trying things out with the quick-and-dirty approach, and said he would try to read the manual and understand the software if he wanted to learn it well. He had earlier provided details on his learning experiences with Microsoft Word's "styles" feature, where he described how the trial-and-error approach had led him to use the program in an inefficient manner.

Another informant (16) noted that he also used trial and error for household appliances, such as a microwave oven. He added that when the approach failed, he usually gave up, and gave examples of his failure to set the time on a car clock or program a VCR.

3.5.4 Strategy 4: Stumbled onto by accident

The phrase, "Stumbled onto it by accident," was intended to cover unplanned learning instances, such as intending to type tab to move the cursor forward but accidentally typing shift-tab and discovering that it moved the cursor back. However, several informants gave examples of other unstructured learning strategies, such as trial-and-error or noticing someone else, both of which fall within a reasonable interpretation of the phrase.

Once they understood the question, informants generally had trouble recalling instances. The only situation in which subjects easily recalled stumbling across new features was when reading manuals, where Informants 4, 5, 8, and 15 had noticed things they weren't looking for. Every one of the other informants said something to the effect of, "I'm sure that must have happened, but I don't know if I can think of an example." Only two informants actually recalled stumbling onto a new feature in an interface: Informant 11 discovered a table-formatting option that was in an inappropriate dialog box, and Informant 12 discovered Microsoft Word's "drag-and-drop" feature. Finally, Informant 9 recalled discovering how *not* to do something in a spreadsheet -- that is, he made an error and the program caught it.

In summary, the question might have been better worded, and it is possible that the informants simply hadn't indexed their experiences for retrieval through the cue provided. But it seems most likely that stumbling across new features in an interface -- and remembering how to use them -- is a rare occurrence.

3.5.5 Strategy 5: Asked someone (in person or by phone)

All informants easily recalled asking for help on computer software problems. The interviews provided weak evidence of a correlation between availability of software help and willingness to ask. The informant most likely to ask for help (15, who said he would "always!" ask for help) was a faculty member in a department with responsive system-support personnel. Three of the four informants who pointed out problems with asking for help (4,11,14) were in situations where support was not so readily available. But no clear measure of willingness to ask or availability could be derived from the interview data.

Several factors were raised that biased users against asking for help. Two informants (11, 14) worked in situations where they were usually the most experienced user of their software, so they could only get help from the phone-support lines, which they both did. Another informant (4), who reported frequently calling a consultant for help, gave the opinion that some expert users were annoyed by the frequent questions from other users. Still another constraint was time, already mentioned in connection with Question 1. Informant 12 stated that he would ask for help if someone was in the room or in a nearby office, but if no one was readily available he would work around the problem, leaving the question to be resolved later when a colleague was available.

An additional factor, pride in one's ability to solve problems, was hinted at in some of the interviews, but the bottom-line analysis indicates that the stereotype of the lone computer scientist may be a false one. Informant 5, a computer science graduate student, summarized the case nicely: "I'm not an asker," he said. "Although, this Ph.D. program has changed that. I ask more questions than I used to. Something about graduating that appeals to me."

3.5.6 Strategy 6: Sent e-mail or posted news request for help

Neither e-mail nor network news programs were widely used by the informants as a problem-solving resource. Only five of the users recalled sending e-mail requests for help. One informant (16) preferred e-mail over phone conversations because it allowed a more detailed communication, but in the end he would select whichever media provided the fastest response. Informant 5 was noncommittal, giving an example of a successful e-mail request to a software author. Informant 11 used e-mail "very seldom," and Informant 8 considered it only an alternative to a phone call. Informant 4 complained that " e-mail doesn't work too well for me, because either I don't explain things too well or people don't pay attention." He added that he would only send e-mail to systems personnel, since sending an e-mail request to a colleague would be insulting.

Three of the users who said they would never use e-mail gave time constraints as the reason. "If a problem comes up, you don't want to wait on it," explained Informant 2. The other users simply stated that they didn't use it or preferred phone calls.

None of the informants recalled posting to network News with a software problem. Four users gave no specific answer to the question. Eight more simply said "no." Informants 3 and 12 said they would never ask a question on News because it was too humiliating.

Informant 3 also thought News was too slow. "I don't want to come back two hours later and get 20 replies." Informant 7's expectations were even lower: "If you post something saying, does anybody know how to do this? Then you get back twelve replies from people saying, yeah, when you find out, tell me. Great. That's really useful."

3.5.7 Strategy 7: Noticed someone else doing it

None of the informants had a quick answer to this question, although 11 of them eventually recalled examples of some sort. Six of the examples involved noticing a specific command given by another user on a computer (2,4,12,14,16), although the interviews did not always reveal whether the informants had noticed the command itself or merely noticed its effect and then asked how to achieve it. Another three (5,7,9,15) clearly recalled noticing the effects, which included a different text previewer, a text format on an overhead slide, a sound output, and a way of combining two forms of data. The remaining two occasions were in the contexts of demonstrations or training sessions, which was not the intended meaning of the question.

Several interesting factors were noted by informants as they considered this question. Informant 2 pointed out that this kind of learning event happened more frequently when his work group had all just started on a new system. Informant 11, who couldn't recall an example, is a faculty member who explained that he almost never worked in an environment where other people were using computers. Informant 15 stated a strong preference for asking for help, and his example of noticing involved noticing that a colleague could do something with the computer, then going back to the colleague at a later date and asking how to do it.

3.5.8 Strategy 8: Other

One informant (4) mentioned user groups as a place to learn things, another (12) suggested videos. Neither recalled specific instances. A third informant (16) mentioned

classes, and several other informants had recalled attending classes when they described their background. Most of the informants, who were reading the categories off a Eureka slip as they gave their answers, simply skipped this category.

3.6 Question: Task-Free Exploration Outside of Computers

Question 4 had established that task-free exploration of computer systems was uncommon for the users studied. We wanted to compare the informants' behavior in the domain of computers to their behavior in other domains. It might be the case that some users are especially inclined to exploratory activities in general, an inclination that might show up in their computer activities. Conversely, if the users did not explore in any domain, then their failure to explore computers might reflect a more fundamental preference or limitation.

To investigate these issues, informants were asked to recall exploratory activities in non-computer domains. Because the words "exploratory activities" would have little meaning to the informants, the question was supplemented by examples: travelling without a detailed itinerary, or shopping with nothing to buy.

All fourteen of the informants readily recalled examples of unplanned, goal-free activities. Travel was the most common domain for exploration. (This may have reflected a bias in the examples given as part of the question.) Other domains included shopping and hobbies, such as gardening.

Eleven of the informants reported exploratory activities as part of their travel experiences. Within this group, however, there was a wide range of exploratory freedom. One informant described several weeks of bicycling through Europe, during which only the trip's end point and one intermediate stopover were preplanned. Each day's activity included a check of the map to see what towns were within biking distance as possible places for the next night's lodging. Another informant described a superficially similar trip: bicycling through Europe for several weeks. However, for this trip all the details had been planned on a spreadsheet, down to the individual hotels and restaurants. Only the activities to fill a small amount of spare time were left unplanned and open to exploration.

All of the exploratory activities described by informants were recreational, providing an obvious contrast to the opinions expressed earlier concerning exploration

of computer systems. In response to that question, several informants had indicated that they did not find it interesting or appropriate to engage in recreational activities with computers, which they considered tools of their work.

3.7 Question: Influence of Exploratory Activities

It had been suggested that exploration, especially in adults, might be rare, but that it might be the source of significant learning events that led to major changes in a person's life. A history of fascination with how machines work might lead someone to study mechanical engineering, for example. To investigate this possibility, informants were asked whether their exploratory activities had ever led to discoveries that significantly influenced their lives.

None of the informants could think of a significant example of exploratory activities that had any significant influence. This negative result is in part related to the answers given to the previous question, about exploratory activities. The activities listed most frequently involved travel and shopping, and informants were hard-pressed to imagine how these activities might have a long-lasting influence.

As a follow-up question, informants were asked how they came to select their current career and how they had made other major decisions leading up to what they were doing at the time of the interview, such as deciding where to go to college or where to live. The answers to these questions provided a rich set of examples of serendipity. Most of the informants were able to recall some unplanned occurrence that had aroused their interest in a topic, or supplied information about a possible choice of university or place to live. But none of these serendipitous occurrences had anything to do with exploratory activities.

3.8 Discussion

Overall, the data collected in the interviews validated and helped to explain the behavior recorded in the daily logs. Most importantly for our research into exploratory learning, the interviews emphasized that users engage in task-free exploratory learning only on very rare occasions. However, they do use exploration as one means to resolve task-oriented difficulties, typically combining trial-and-error investigations with the use of manuals and interactive assistance from other users or system support personnel.

4. FINAL DISCUSSION AND SUMMARY

As described in the introduction, earlier research had suggested that exploratory learning could be a productive and enjoyable strategy for learning about computer applications. However, there was reason to believe that unsupported instructionless learning, without resort to manuals or other training aids, would be problematic. The goal of the diary logs was to observe users' learning strategies in the work place, under the constraints of their daily jobs, with particular attention to strategic use of available resources. The interviews helped to validate the logs and extended the study's coverage to include strategies for initially learning new computing systems.

4.1 Methodological Comments

The diary logs combined with interviews proved to be a valuable research tool. The raw quantitative data yielded by the logs was supported by "softer" reports during the interviews, and the data as a whole was informed by the understanding that arose during the personal interactions between the investigator and the informants. The mixture of hard and soft data in field methodologies such as this forms a critical bridge in HCI research, allowing the investigator to validate the ideas derived from anecdotal reports before committing to tightly controlled laboratory studies (Rieman, 1993). This approach to improving the contextual validity of laboratory research is one answer to the problems identified by proponents of situated cognition and related design methodologies (Suchman, 1987; Wixon, Holtzblatt, and Knox, 1990).

In the study reported here, the investigator's ability to understand the data and to communicate in the informant's own terms during the interviews was largely the result of the rapport developed during the daily debriefings. These personal interactions were also important in gaining and holding the confidence of the informants, who were committing a fair amount of their time and who would inevitably reveal some personal opinions and activities during the course of the logs and the interview. On the down side, the close interactions also made it more likely that informants would adjust their activities to please (or displease) the investigator, so it was especially important that the logs and Eureka slips were not biased toward any specific learning strategy.

As with any empirical research, the results of the study must be understood as limited to the population from which the informants were drawn. The study investigated the behavior of a relatively well-educated, computer sophisticated group of users, most of whom have had long-term access to modern computing facilities, including display-based personal computers and electronic mail. The behavior of these users, in particular their reliance on manuals and help from support personnel and other users, reflects both the information richness of their environment and the confidence and sophistication of the individuals. Nardi (1993) and MacKay (1990) found similar collaborative work in studies of spreadsheet and CAD users. The results of this research, therefore, describe a situation that may be widespread in the culture of today's workers in large organizations, but that may not apply to individuals who work at home or in very small offices. The situation within large organizations may also change as computer systems and their associated resources continue to evolve.

4.2 Summary of Results

The results of the study confirmed the value of exploratory learning and provided important details concerning its natural occurrence. It was found that informants used a variety of methods to gain a basic competence with their systems, including manuals, task-oriented on-line exploration, informal use of tutorial materials, and demonstrations by other users. They supplemented the skills gained in this initial phase by further learning as work demanded. Their strategies for ongoing learning most often combined on-line trial-and-error investigations with the use of printed manuals and, less frequently, with interactive help from other users and systems personnel. Without exception, informants in the study were willing to engage in at least some amount of exploratory behavior when a real-world task took them beyond the bounds of their current computing skills. On-line help, while used occasionally by some of the informants, was widely disliked. Electronic mail and network newsgroups were almost never used to resolve problems.

Throughout the study, behavior and interviews repeatedly showed that informants were strongly concerned with the constraints on their time. They often selected initial learning strategies that would postpone full learning of a package until a

real task made it clear that the investment of time was worthwhile, and they preferred problem-solving resources that were quick and immediate.

For informants of all skill levels, the study revealed a paucity of task-free computer exploration. In their behavior and their interview statements, the majority of users indicated that they believed such activities to be time-consuming and unproductive. For most of the informants, current computing environments do not have the intrinsically motivating nature of computer games, and these users evidenced little or no desire for such a characteristic in their work-related software.

4.3 Implications

The results of the study have application in several areas. The reliance on the expertise of other users and system support personnel, often as a critical fall-back resource when manuals and on-line exploration fails, emphasizes that collaborative behavior is an important factor in successful computer use. System designers should recognize this activity, which might be termed "CWCS" (cooperative work on computing systems), and be prepared to support it with extra documentation and auxiliary programs for systems personnel or key users. Managers, also, can benefit from the observation that a productive environment is likely to include support personnel who respond rapidly to help requests, as well as a cooperative work attitude that encourages users to spend a few minutes sharing their skills with others when those skills are needed.

Turning to the details of the learning strategies, the informants' general dissatisfaction with on-line help raises the question of why this technology so often fails, and how it might be improved. One possible answer is suggested by Table 8. Seventeen out of 26 instances of reading the manual, and 8 out of 9 instances of using on-line help, were combined with trying things out in the interface. For many systems, however, trying things out and using on-line help cannot be done concurrently. This is not the case for paper manuals, since they can be left open while the system state is altered and observed. A second possibility, implicit in several of the interviews, is that on-line help systems do a poor job of handling the distinction between high-level overviews and low-level details. Overview information includes facts such as the existence of a feature within a software package, while detailed information might give the name of the menu

under which a feature is found (Lewis & Rieman, 1993). A human helper can usually identify the level of information needed, even if the wrong level is requested; and some users have developed scanning skills with manuals that yield similar results. But for several of the informants in this study, on-line help systems promised only to overwhelm them with information at the wrong level.

Finally, for the purposes of further research into exploratory learning, the study describes the context in which such learning typically takes place. Task-free exploration is unlikely to occur, not only because systems are uninteresting, but because users' fundamental goal is to complete their work-related tasks, and they prefer quicker, more focused learning methods to that end. Task-oriented exploration, however, is common, and its structure is shaped by the existence of a real, work-related task, significant time constraints, and multiple fall-back resources that will be turned to if on-line investigations fail. These observations have helped to shape recent research in our laboratory (Franzke & Rieman, 1993; Franzke, 1994; Rieman, 1994), and they raise important questions concerning the low-level details of resource selection, use, and interaction during task-oriented exploration.

ACKNOWLEDGMENTS

I thank Clayton Lewis, Peter Polson, and Marita Franzke for their advice and collaboration on the research effort of which this is a part. Anders Ericsson provided suggestions as to the form of the diary logs. Fourteen anonymous informants volunteered their time for the diary logs and interviews; their cooperation is greatly appreciated.

REFERENCES

Carroll, J. M. (1982). The adventure of getting to know a computer. IEEE Computer 15 (11), 49-58.

Carroll, J.M. (1990). *The Nurnberg Funnel.* Cambridge, MA: MIT Press.

Carroll, J.M., Mack, R.L., Lewis, C.H., Grischkowsky, N.L., and Robertson, S.P. (1985). Exploring a word processor. *Human Computer Interaction, 1,* 283-307.

Carroll, J.M., and Mazur, S.A. (1986). Lisa Learning. *IEEE Computer 19* (10), 35-49.

Carroll, J.M., and Rosson, M.B. (1987). The paradox of the active user. In J.M. Carroll (ed.), *Interfacing thought: Cognitive aspects of human-computer interaction.* Cambridge: MIT Press/Bradford Books, pp. 80-111.

Engelbeck, G.E. (1986). Exceptions to generalizations: Implications for formal models of human-computer interaction. Unpublished Masters Thesis, Department of Psychology, University of Colorado, Boulder, CO.

Ericsson, K. Anders, Tesch-Römer, C., and Krampe, R.T. The role of practice and motivation in the acquisition of expert-level performance in real life: An empirical evaluation of a theoretical framework. In *Encouraging the Development of Exceptional Skills and Talents,* M.J.A. Howe, Ed., The British Psychological Society, Leicester, 1990, pp. 109-130.

Fischer, G. (1987). Cognitive view of reuse and redesign. *IEEE Software, Special Issue on Reusability, 4* (July 1987), 60-72.

Franzke, M. (1994). Exploration and Experienced Performance with Display-Based Systems. Ph.D. Dissertation, Department of Psychology, University of Colorado.

Franzke, M., and Rieman, J. (1993). Natural training wheels: Learning and transfer between two versions of a computer application. *Proceedings of Vienna Conference on Human Computer Interaction 93.* Sept. 20–22, 1993. Vienna, Austria.

Lewis, C. (1988). Why and how to learn why: analysis-based generalizations of procedures. *Cognitive Science,* 12, 211-256.

Lewis, C., and Rieman, J. (1993). *Task-Centered User Interface Design: A Practical Introduction.* Boulder, Colorado: Shareware electronic publication, available via anonymous ftp to ftp.cs.colorado.edu.

Mackay, W. (1990). Users and customizable software. A co-adaptive phenomenon. Ph.D. dissertation. Sloan School of Management. MIT, Cambridge, MA.

Malone, T. W. (1982). Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games. *Proceedings of the Conference on Human Factors in Computing Systems* (Gaithersburg, MD). New York: Association for Computing Machinery, 63-68.

Nardi, B. (1993). *A Small Matter of Programming.* Cambridge, Mass: MIT Press.

Neal, L.R. (1987) Cognition-sensitive design and user modeling for syntax-directed editors. *Proceedings of CHI+GI '87 Conference on Human Factors in Computing Systems and Graphics Interfaces,*. New York: Association for Computing Machinery, 99-102.

Nielsen, J., Mack, R.L., Bergendorff, K.H., and Grischkowsky, N.L. (1986). Integrated software usage in the professional work environment: evidence from questionnaires and interviews. *Proceedings of CHI'86 Conference on Human Factors in Computing Systems,*. New York: Association for Computing Machinery, 162-167.

Polson, P.G. and Lewis, C.H. (1990). Theory-based design for easily learned interfaces. *Human-Computer Interaction, 6,* 191-220.

Rieman, J. (1993). The diary study: A workplace-oriented tool to guide laboratory studies. *Proceedings of InterCHI'93 Conference on Human Factors in Computer Systems.* New York: Association for Computing Machinery. pp.321–326.

Rieman, J. (1994). Learning Strategies and Exploratory Behavior of Interactive Computer Users. Ph.D. Dissertation, Department of Computer Science, University of Colorado.

Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages. *IEEE Computer,* 16 (8), 57-69.

Shrager, J. (1985). Instructionless learning: Discovery of the mental model of a complex device. Unpublished Ph.D. dissertation, Carnegie-Mellon University.

Shrager, J., and Klahr, D. (1986). Instructionless learning about a complex device: The paradigm and observations. *International Journal of Man-Machine Studies, 25,* 153-189.

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

Wixon, D., Holtzblatt, K., and Knox, S. (1990). Contextual design: An emergent view of system design. In *Proceedings of the Conference on Human Factors in Computing Systems* . New York,: Association for Computing Machinery, 331-336.