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. These findings have implications for the design of systems, documentation, and training.

Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques—*user interfaces*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*training, help, and documentation*

General Terms: Documentation, Human Factors

Additional Key Words and Phrases: Diary studies, discovery learning, exploratory learning, learning in the workplace, learning on demand

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 had been formally trained in their

This work has been supported by grant number IRI-9116640 from the National Science Foundation, with additional work funded by the UK Joint Councils Initiative. A preliminary report of this study was presented at INTERCHI ’93.
Authors’ address: Department of Computer Science, University of Colorado, Boulder, CO 80304; email: rieman@cs.colorado.edu.

use. Learning to operate the word processor in the office where the author worked required two days of classroom training, and employees who had not completed the course were not 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 was not 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 is not 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 and 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, et al. 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 article is to put that anecdotal observation on firmer footing and to discuss its implications for system design and further investigations.

1.1 Studies of Exploratory Learning

The potential value of exploratory learning has been recognized since interactive computing first began to enter the workplace. 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 et al. 1985]. The research demonstrated that most users were willing and able to learn by exploration, although there were individual differences in exploratory aggressiveness [Carroll 1990; Carroll and Rosson 1987]; see also Neal [1987]. It also showed that novices attempting to explore a totally new system often made 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] and 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; however, 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 and 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, which is related to the identity heuristic in Lewis' [1988] EXPL theory. In label following, users take the afforded actions on controls whose labels are identical or similar to words and concepts in their current task. More recently, Franzke [1995] 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 field study described in this article 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 need for their daily tasks?

In brief, the study had two parts. First, diaries were maintained for one work week by 14 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. Second, following the diary studies, a structured interview was performed with each of the same users. 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. Instead of investigating novices' first experiences with systems, the study observed users with a range of experience as they extended their knowledge of systems they worked with every day. Instead of performing a controlled comparison between types of manuals or training, or observing users forced to explore with no manuals at all, the 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 are potentially valuable in several areas. Most obviously, software designers should be able to craft more effective systems, including manuals and on-line documentation, if they know how users learn software on the job. This will be especially true if the designers can understand why the work environment causes a user to prefer a specific resource for learning, such as manuals, on-line help, or exploration. A similar argument holds for the design of training: a knowledge of users' ongoing learning behavior helps define the need for initial baseline training. Finally, the study's broad description of exploratory behavior can help to guide more controlled laboratory experiments into specific issues [Franzke 1995; Franzke and Rieman 1993]. As such, the study reflects one 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 1994].

1.3 Organization

The rest of this article is organized as follows. Section 2 describes the diary study and its results, including a detailed discussion of the diary method and a description of the users who participated in the study. Section 3 describes the results of subsequent in-depth interviews with the same users. Section 4 concludes by summarizing the findings and discussing their implications for training, systems design, and further research.

2. THE DIARY STUDY

The diary study recorded the activities of 14 interactive computer users in the course of one week of their ordinary work. The study was designed to show the extent to which users learned new features about their systems outside of a training situation, along with the strategies and resources used during that learning.

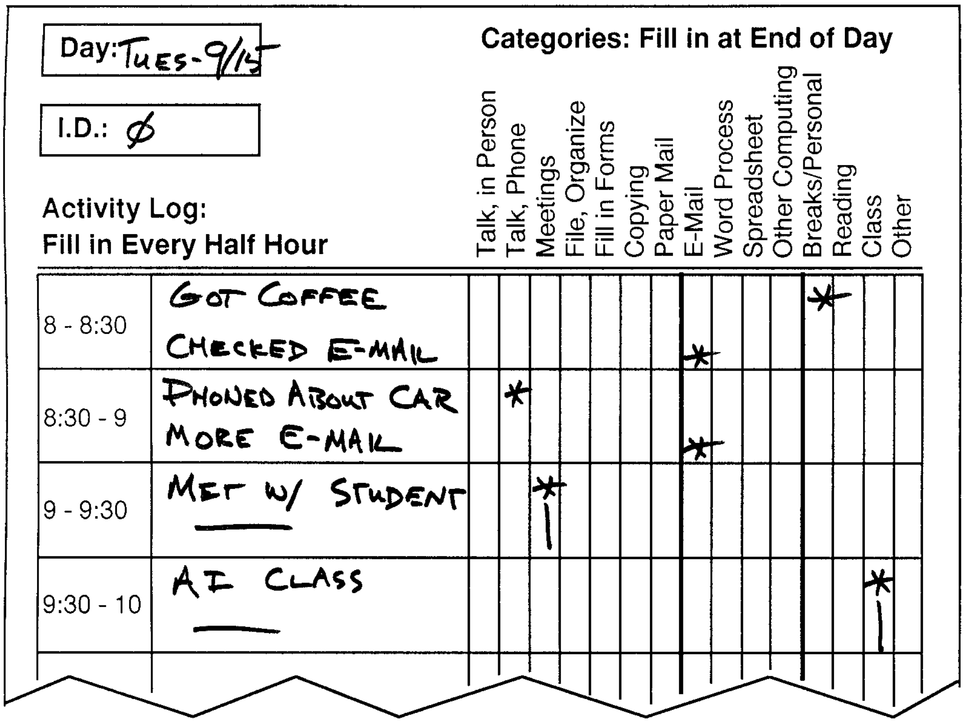
2.1 Diary Method

In diary studies [Ericsson et al. 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 strength of the diary study is its ability to provide data about real-life activities in the workplace. When combined with a follow-up interview, the method promotes the collection of a structured set of core data while providing an easy opportunity to recognize and collect additional information. However, diaries and interviews both require that informants self-report their activities after the fact. There are potential dangers with this kind of self reporting. Informants may fail to recall all activities of interest, so the report could be incomplete. Informants' beliefs about the study's purpose may bias their selection of events to report or their descriptions of those events. Informants could even adjust their behavior to produce the activities that were apparently desired. The materials and procedures were designed to avoid these problems as much as possible. In addition, the use of a diary study followed by the interviews provided an opportunity to acquire converging data on key questions.

2.1.1 *Materials.* Informants maintained their daily log of activities on the form shown in Figure 1. The form was on 11-by-14-inch paper, and the hours on the left side corresponded to the informant's typical working hours, with a few hours 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 categories on the log and on the Eureka slip were initially selected based on anecdotal evidence of typical work activities and learning strategies and were redefined after a small pilot study.

The field study was specifically designed to identify the existence and context of learning events, without clearly exposing their structure. Consequently, the grain size of the data solicited by these materials was relatively large. Events that took several minutes could be reported, but the low-level detail of those events would not be described. A log that reported a more detailed temporal structure of learning events could have been designed, but the burden of producing detailed reports might have



Day: TUES-9/15

I.D.: Ø

Activity Log: Fill in Every Half Hour

Categories: Fill in at End of Day

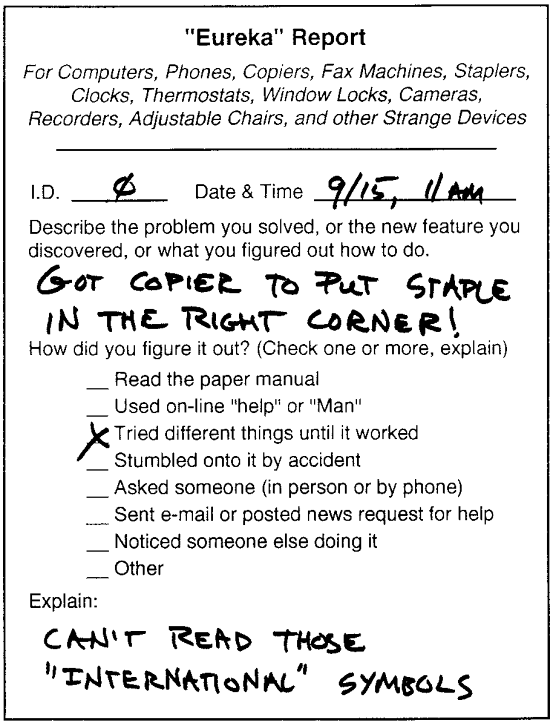
		Talk, in Person	Talk, Phone	Meetings	File, Organize	Fill in Forms	Copying	Paper Mail	E-Mail	Word Process	Spreadsheet	Other Computing	Breaks/Personal	Reading	Class	Other
8 - 8:30	GOT COFFEE CHECKED E-MAIL								*				*			
8:30 - 9	PHONED ABOUT CAR MORE E-MAIL		*						*							
9 - 9:30	MET W/ STUDENT	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

Fig. 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.

discouraged informants from reporting every event. (John and Packer [1995] describe a diary study that focused on the structure of learning, but they logged only learning events, not entire work days.)

Several features of the materials addressed the problems of self reporting. The focus of the study was learning, specifically exploratory learning, but the daily log sheets helped to avoid bias by asking for a full report of the day's activities. The sheets provided a special section for computer activity, but no category for learning. The Eureka slips narrowed the focus to learning, but they were broad enough to cover many kinds of learning activity, including both task-oriented and task-free exploration. The bright colors and ready accessibility of the Eureka slips were intended to make them clearly visible and increase the likelihood that informants would remember to fill them in.

2.1.2 *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 traveling 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.



"Eureka" Report

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

I.D. Ø Date & Time 9/15, 11 AM

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 THE RIGHT CORNER!

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

__ Read the paper manual
__ Used on-line "help" or "Man"
[✓] 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 READ THOSE "INTERNATIONAL" SYMBOLS

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

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, by phone for Informants 9 and 12). 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 sometimes made additional notes on the back of the slips or corrected a strategy assignment.

The end-of-day meetings helped to regularize reporting among informants. The informants were willing but often busy, and the daily collection of log sheets ensured their daily completion. 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. The meeting also provided an opportunity for the informant to ask for clarification about the reporting procedure; questions as to what constituted a "Eureka" event were common on the first day or two. These daily 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. However, 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 minimally biased.

2.1.3 *Informants.* Fourteen informants participated in the study, seven male and seven female. They are identified in this report as Informants 1 through 14, and are all referenced with male pronouns to help preserve anonymity. All participants volunteered their time.

The informants were selected to provide a broad range of experience and work demands. This section provides details as well as summary discussion of the informants' backgrounds and work situations during the diary period, since these are key factors in determining the broader applicability of the study. This information was collected as part of the first question in the interview phase of the study. Table I provides a brief description of each informant.

The informants' current use of computers was a key factor in their being asked to participate in the study. Four of the informants, identified by the key word "clerical" in the computer-use column of the table, represented the large population who use word processors and spreadsheets in an office environment. Another three informants, categorized under "science support," were selected to yield insights into the work of scientists who rely on computers for text processing, graphics, and some data analysis, but are not involved in computer modeling or other research that is fundamentally computer based. The three informants whose computer use is described as a "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 these informants worked in business, where they did financial modeling and analysis using interactive spreadsheets and databases. The third informant in this group was a cognitive scientist who used computers as a tool for cognitive modeling. Finally, four of the informants were highly experienced computer scientists, categorized as performing "work/research" in the field of computers themselves.

The informants' experience with computers was also distributed across a very wide range, in terms of both breadth and depth. Expressing this range in units such as "years of experience" or even in broad categories such as "novice" or "expert" is difficult. However, to provide an overview of the sample characteristics, informants could be roughly assigned to three groups. Six informants had experience as users that ranged from a single word processor, email, and one other application, to several applications

Table I. Informants' Background Experience

ID	Operating Systems*	Computer Use	Position; Duties	Background and Experience
1	Unix/ Workstation	clerical	secretary; general	On-the-job and classroom training on Unix-based and dedicated word processing, spreadsheets.
2	Mac/VMS	clerical	secretary; general	On-the-job training, some classes, on dedicated word processors, PC and Macintosh-based word processors and spreadsheets.
3	Workstation/ 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.
4	Mac/Unix	work/research	Ph.D. student in computer science; AI research, writing	Industry experience in program development and support. Experience with Unix, PCs; learning Macintosh. Has taught computer science courses.
5	Mac/PC	clerical	undergraduate in engineering, clerical assistant duties; homework	Home experience with PCs. Work experience with word processing and spreadsheets on Macintosh. Some programming training.
6	Unix/ Workstation	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.
7	Mac/Unix	primary tool	research faculty in cognitive science; cognitive modeling, empirical research	Master's degree in computer science, Ph.D. in cognitive science; has taught computer science courses. Macintosh user and Lisp programmer. Research focus in psychology.
8	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.
9	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.
10	PC/VMS	science support	faculty member in linguistics; 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.
11	Mac/VMS	science support	Ph.D. student in psychology; empirical and 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.
12	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 PCs and Macintosh, both as a user and support person. Frequent programmer in database and spreadsheet packages.
13	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 PCs. Email on minicomputer. Has avoided learning to program.
14	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.

* Primary/secondary operating system; PC includes both MS-DOS and MS Windows.

Table II. Hours Logged During the Diary Study

	Total	Mean	SD	Min	Max
Hours Logged, Less Personal Time	476.5	34	4.3	27.0	39.5
Hours of Computing	242.6	17	8.4	1.0	34.4
% of Log Hrs Spent Computing	51	51	24	3.5	91.7

and systems. None of these informants programmed as a regular part of their work. Another four informants had work experience with many applications, some responsibility for the support of other users, and programmed frequently, either in traditional languages or in 4GL systems. The last four informants were professional programmers in research or industry, who had formal training and had taught or supported other users.

Informants also used a variety of computer operating systems as their primary system. Systems listed as "secondary" in Table I are those with which informants had significant past experience or current access for occasional use, typically for email purposes or as a home system. For some uses it was difficult to say which was the primary system; in these cases the system listed in the table is the one in which the user reported the most computing during the log week.

2.2 Diary Results: Logged Time

This section presents a summary of the log sheet data. These data do not report learning events, which were recorded on the Eureka slips. To a large extent, the purpose of the log sheets was to draw the informant's attention to the diary process, so Eureka events would be recorded when they occurred. However, the log sheet data is also of interest because it shows the amount and character of computer interaction within which learning events occurred.

2.2.1 *Total Time and Time Computing.* The data presented include only logged time not marked as breaks/personal. All but two of 14 informants kept their logs over a consecutive five-day period. One informant (2) kept a log over 4.5 consecutive work days plus a half day a week later; for another (9), 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.

The total time logged and time spent computing is given in Table II. Informants spent about half their time, on average, working with computers (17 hours out of 34). However, the range for individuals extended from 2.5% (Informant 5, 1 hour out of 28.5) to 92% (Informant 7, 27.75 hours out of 30.25). For the purposes of this analysis, computers were defined narrowly to include PCs, minicomputers, and mainframes. They did not include phone systems, VCRs, fax machines, copiers, and similar hardware

Table III. Hours Logged in Software Categories During the Diary Study

	Total Logged			Percent of Informants' Computing Times			
Software Category	Hours	% Ttl	Infs*	Mean	SD	Min	Max
Word Processing	88	36	12	38	31	0	89.5
Programming	29	12	4	8	26	0	97.3
Email	26	11	9	15	22	0	60.2
Database	26	11	4	8	14	0	40.6
Special Applications	24	10	4	14	27	0	85.9
Spreadsheets	17	7	5	6	15	0	55.2
Graphics	12	5	4	4	9	0	33.9
Telecom (up/dnload)	9	4	4	2	5	0	16.7
Operating System	7	3	7	3	6	0	21.0
Games	4	1	1	1	5	0	20.3
News/On-Line Info.	3	1	3	1	3	0	8.7
Unknown	<1	<1	1	<1	<1	0	1.7
Total	243		14				

* Number of informants who logged time in the category.

that typically has a dedicated internal microprocessor and a multifunction interface.

2.2.2 *Time in Application Categories.* As part of the log, informants recorded the category of computer application they were working with. The data are summarized in Table III. 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. The World Wide Web was not available to these users when the study was performed.

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% of the total computing hours logged. No other category stands out.

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 38%, but with a very great range. Two of the 14 informants did no word processing at all. Of the remaining categories in Table III, only electronic mail shows a mean percentage (15) and a number of informants (9) that even approaches a consensus.

Table IV. Hours and Eurekas Logged During the Diary Study

	Total	Mean	SD	Min	Max
Hours of Computing	243	17	8.4	1	34
Number of Eurekas	60	4.3	4.0	0	15
Eurekas per 8–Hr Computing (E/8hr)	1.7	1.7	1.3	0.0	4.1

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 11 spent 15 minutes browsing the university's on-line information system with no set goal, but this was an exploration of information, not of the interface. Informant 10, who had recently acquired new software, spent more than five 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 was not great enough to justify the time he was spending with it. Three other informants (4, 9, 14) were working with new software during the log week, but all of them structured their activities entirely around real, current tasks.

2.3 Eureka Slips

The Eureka slips recorded the learning events that the informants reported during the log period. This data was analyzed to investigate the rate at which learning events occurred and the strategies used for learning. A total of 78 events were recorded. Of these, 18 involved copiers, phone systems, or other equipment whose operation was not included in the calculation of each informant's "computing time." Those 18 Eurekas are not included in the detailed analysis that follows.

2.3.1 *Eureka Distribution by Informant Category.* The distribution of the 60 computer-related Eurekas across informants is summarized in Table IV. One informant (4) reported 15 Eurekas; the counts for the other informants ranged from 0 (Informants 5 and 13, who both reported noncomputer Eurekas) to 8 (Informant 10). The total hours spent in computing for each informant has been used to normalize the Eureka counts, 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.

Table V. Eurekas per 8-Hour of Computing, by Informant Categories

	Mean	Mean*	Values
Overall	1.7	1.2	
by Gender:			
female	2.0	1.3	
male	1.5	1.1	
by Computer Use:			
clerical	1.2	1.2	(1.8, 2.3, 0, 0.62)
science support	2.1	1.1	(4.1, 2.2, 0)
primary tool	2.1	1.4	(1.7, 1.1, 3.4)
work/research	1.7	1.1	(1.6, 3.5, 0.9, 0.9)
by Primary Operating System:†			
Macintosh	1.5	1.1	(2.3, 3.5, 1.7, 0.62, 2.2, 0, 0)
MS-DOS/MS Windows	2.9	1.1	(4.1, 1.1, 3.4)
Unix/VMS	1.2	1.2	(1.8, 0.9, 0.9)
Workstation	1.6	1.6	(1.6)

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

The mean E/8hr score for all informants is 1.7, but this is strongly influenced by the relatively high scores for three users: Informants 4 (with a raw Eureka count of 15), 10, and 14 had scores of 3.5, 4.1, and 3.4, respectively. Not surprisingly, all of these users were actively involved in learning newly acquired systems during the diary week. The E/8hr scores for all other informants are less than or equal to 2.3, with a mean of 1.2 (SD = 0.8). By far the most prevalent case, therefore, is that of the user who learns new things about a system infrequently, perhaps only two or three times a week in an environment where spending half of one's time in computing activities is common.

In Table V, the E/8hr scores of informants are broken down by gender, 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.

Because of the difficulty in assigning experience ratings, no firm correlations can be made between experience and E/8hr scores. An analysis using the informal categories shown in Section 2.1.3 suggests that the number of reported Eurekas did not change noticeably with experience. However, examining the Eureka slips shows that novice users recorded many simple Eurekas, such as learning how to select several files at one time on a Macintosh; while more experienced users sometimes recorded complex Eurekas, such as getting a simple "hello-world" program to run in a new database language.

2.3.2 *Eureka Distribution by Strategy.* A central purpose of this research was to identify the techniques that users apply when learning about their systems. Table VI presents the Eureka data that are most relevant to this issue. The table shows how often each of the strategies on the Eureka

Table VI. Distribution of Eurekas by Strategy

		Strategies Checked*							
	Total	Try	Read	Ask	Help	Stmbl	Notice	Email	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 Eureka slips listing the category; some slips listed more than one category (see Figure 2 for full text of strategies); † several user-defined strategies, no more than 2 Eurekas using any one strategy.

slips were used to solve a problem. 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 appears to be generally useful. The relative importance of these learning resources is a key result of this study, and it will be supported by the interviews described in Section 3.

Table VII 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. More than 25% of the Eurekas 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 VI as the most common strategies, are also the only strategies likely to resolve a problem without recourse to some other method. It is especially interesting that on-line help is never successful alone.

2.4 Key Results from the Diary Data

The logged time shows a sample population that is using word processing roughly one third of the time, with the remaining time distributed across many other applications. Within that environment, learning occurs infrequently, perhaps only one learning event for every eight hours of computing time for a user who is not actively involved in learning a new application. Because the data is self-reported, this finding could be lower than the actual value. The complexity of learning events varied across informants, so some learning events may reflect the acquisition of more facts than others.

The data are stronger when viewed as a sampling of learning events that did occur, without regard to whether all such events were recorded. Across a very wide range of situations and users, the Eureka reports showed that the three preferred strategies for resolving problems are trying things out, reading the manual, and asking for help. On-line help is occasionally used, but other strategies are infrequent.

Table VII. Combinations of Strategies Shown with Eureka Reports

		In Combination with. . .*							
	Alone	Try	Read	Ask	Help	Stmbl	Notice	Email	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	—	—	—	—
Email	—	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 to 1 to try-ask, 1 to try-read, and 1 to read-ask.

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 investigator interviewed each of the 14 informants individually, 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.

3.1 The Interview Method

The interviews were structured around a series of prepared questions (Table VIII). The questions covered essentially the same topics that had been the focus of the log period, but with some extensions to exploratory behavior outside the workplace. The investigator would follow up informants' initial answers on each question by encouraging them 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.

The interviews were designed to be complementary to the diary study in several ways. The diary logs yielded quantitative results describing behavior in the short term, while the interviews extended that result with more qualitative descriptions of longer-term activities. The two methods also probed for similar data in different ways, providing multiple retrieval opportunities. For example, the logs asked informants to report learning events as they occurred, then to categorize them as to strategy; in the interviews, an alternative retrieval cue was strategy, to be exemplified by events (Question 5). Finally, the daily debriefings in the diary study served to lay a foundation for the interviews. They created a rapport and a common vocabulary that enhanced the investigator's ability to communicate in the informants' own terms.

All but one of the interviews were taped and transcribed by the investigator. The transcriptions and notes taken during the interviews were

Table VIII. Questions Asked in the Structured Interview

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 traveling, or take apart things just to see how they work?

The wording of each question varied slightly from one informant to another.

examined to produce the summaries in this section. (The answers to the first question, on background, were summarized in Section 2.)

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 *Five Approaches to Learning.* The informants identified five 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: 2–4, 7–11),
—exploring its functionality, usually in the context of actual tasks (7 users: 3–6, 9, 10, and 12),
—working through the supplied tutorial materials (6 users: 4, 5, 10–12, and 14),
—having an experienced user demonstrate the package (5 users: 1, 2, 7, 8, 13),
—learning the basics initially, then learning more as the task demands (5 users: 8–11, 14).

In addition to the five main strategies, a few users (1, 2, 8, 12) had taken classes on some occasion, but this was always an exception to the way they expected to learn a program. One user (11) 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. At other times they 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 (5, 8, 11) described their approach to learning a new piece of software in terms of a few past instances, which had offered different learning options. 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 specific approach to using the mixture of documentation that has become fairly standard in that arena. Informant 10, for example, always started with the video if there was one, then followed the installation instructions and did the tutorial, then tried small sample tasks, then turned to larger, real tasks. Other informants had slightly different approaches.

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 4 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 9, 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 3 made a similar distinction between simple and complex packages, stating that reading and understanding, including an understanding of the source code, were 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 that they performed this exploration in the context of tasks. One of these informants (9) performed both task-oriented and task-free exploration.

For most users, the most effective approach was to use their own tasks. Informant 4 used his own tasks because in "demand-driven" mode he would not waste time learning what was not needed. Informant 10 would sometimes begin exploration with simple, trial tasks, but would soon progress to larger, real tasks. Informant 14 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 3 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 *Use of Manuals in Initial Learning.* Informant 14 expressed the consensus opinion of the value of manuals when first investigating a system: "Reading the user's guide is, to me, useless. I only use the user's guide when I want to do something specific." Eight informants did mention using manuals while learning new software, but the manual was generally used as a supplement or a support to other strategies, such as on-line exploration or demonstration by another user. Only three informants (3, 4, 9) 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 application, such as a programming environment.

3.2.5 *Time Constraints.* Many of the users explicitly identified time as a constraint on their learning activities. Informant 1 did not 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 14 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 2 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 6 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 4, 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 it unsuccessfully 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 his current task in the word processing package he was 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 working through on-line tutorials where available, reading the manual, 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 the first strategy for 9 users (2–5, 10–14). An additional 2 users (2, 7) 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 (1, 8, 9) and a second strategy for 7 (1, 3, 5, 10–12, 14).
—Asking for help was a first strategy for 2 users (2, 6), a second strategy for 5 (1, 5, 7, 8, 13), and the third strategy for 4 (3, 9, 10, 12).

Additional strategies identified were working around the problem (Informants 3, 4, 9, and 11, the first three of whom would sometimes use programming for the workaround—although Informant 3 also praised the efficacy of white-out), using on-line help (Informants 4, 6, and 10) and looking at the source code (a first strategy in complex systems for Informant 3, and a second strategy, after asking colleagues, for Informant 10).

Some informants identified alternate "first" and "second" strategies, depending on the problem, time, and resource availability, and in two cases (1, 2), their mood at the time the problem arose. According to Informant 1, "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 as systems support.

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 (10) who identified this as a common activity had, during the diary week, spent one-third of his computer time exploring new software. Other examples he gave of exploratory behavior were the changing of 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 need 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 was not productive.

One other informant (8), an undergraduate, said he liked to do task-free exploration, but did not have much opportunity. "[For] 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 coworkers.

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 4, and then gave the details of his experience with a Mandelbrot set program. Four of the informants (3, 7, 9, 12) 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 (2, 3, 5–7, 12, 13) specifically said they did not have the time or felt that exploration was not productive, at least not in their current job.

The second clearly expressed reason 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 13. A similar attitude was expressed by Informant 7: "I don't explore the actual computer all that much. Of course, the computer's a tool." And Informant 11, 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 were largely validated by the Eureka counts. It is helpful to understand the reasoning behind the users' preferences. Were the preferred strategies the only ones available? Or were other strategies avoided because past experience had shown them to be inefficient or unlikely to succeed?

To investigate these issues, and to provide the informants with a different index into their learning experiences, 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 preferred strategies discussed with users earlier in the interview, such as the use of manuals, the answers

Table IX. 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* 1, 4, 8, 12	*yes, but problems* 3, 6, 7, 9, 10, 11, 14	*tried, doesn't work* 2, 13	*no, no comment* 5
Tried things until it worked	*yes, de novo* all but three	*yes, with manual* 7, 8, 9		
Stumbled onto by accident	*yes, in interface* 8, 10, 11	*yes, in manual* 3, 4, 7, 13	*maybe, can't recall* 1, 2, 5, 6, 9, 12, 14	
Asked in person or by phone	*colleagues or system support* all			
Sent email	*unqualified yes* 4	*yes w/reservations* 3, 7, 10, 11	*no, too slow* 1, 2, 11	*no, no comment* 5, 6, 8, 12, 13
Posted to network news			*no, for reasons* 1, 2, 3, 6	*no, no comment* 5, 7, 9, 11, 13*
Noticed someone else	*yes, serendipitous* 1, 3, 4, 6, 7, 11, 12, 13, 14		*only in training or demo* 5, 7	*no, no comment or can't recall* 2, 10
Other	*yes* (*user group, video, class*) 3, 10, 14			*no, can't think of any* 4, 8, 13**

* Informants 4, 8, 10, 14 did not answer the "net news" question; ** Informants 1, 2, 4, 6, 7, 9, 11, 12 did not answer the "other" question.

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 IX, and additional details are given below.

3.5.1 *Strategy 1: Read the Paper Manual.* Manuals had been mentioned by most informants during earlier questions, and they all recalled instances of solving problems by reading a manual, without much further discussion. For some users, the most useful manual was a commercial "How-to-Use-Program-X" text, written by someone other than the software manufacturer. Additionally, 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 the Unix system's man, but for the majority of them it was not highly regarded. 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 13, explaining that he either did not understand the help message or else that it clearly did not answer his question. Informant 2 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."

Seven users recalled instances of finding things using on-line help, but qualified their opinion of the approach. Informant 10 would use on-line help on DOS systems, but only if the manual was not readily available. Informants 3, 7, and 11 would use on-line help for command-oriented systems (Unix, VMS, SAS), but not for the software they had used on the Macintosh. On the other hand, Informants 9 and 14 would use on-line help with the Macintosh, but not with command-line systems (Unix and DOS, respectively). Comments on Unix man, even by informants who used it, were negative but not specific as to the problems. Informant 6, for example, stated that "it doesn't usually amount to much."

Only one user (Informant 4, a professional with experience in DOS, Macintosh, and Unix) described on-line help (typically in DOS) as his first fallback after trying things that had failed.

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 was not a lot of further discussion. Three of the informants (7, 8, 12) specifically noted that they often used trial and error to disambiguate information from the manual. (Informants 7 and 2 had made the same comment in response to Question 1.)

One of the highly experienced informants (9) noted that this approach was common in programming, where he defined it as "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.

3.5.4 *Strategy 4: Stumble onto It by Accident.* The phrase "stumble onto it by accident" was intended to cover unplanned learning instances, such as intending to enter the tab key to move the cursor forward but accidentally entering shift-tab and discovering that it moved the cursor back. Informants generally had trouble recalling instances of this behavior. The only situation in which subjects easily recalled stumbling across new features was when reading manuals, where Informants 3, 4, 7, and 13 had noticed things they were not looking for. Only two informants actually recalled noticing an unsought feature in an interface: Informant 10 discovered a table-formatting option that was in an inappropriate dialog box, and Informant 11 discovered Microsoft Word's "drag-and-drop" feature.

3.5.5 *Strategy 5: Ask Someone* (*In Person or by Phone*). All informants easily recalled asking for help on computer software problems. However, the availability of software help was clearly an issue. Informant 13, 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 by asking for help (3, 10, 12) were in situations where support was not so readily available.

Several factors were raised that biased users against asking for help. Two informants (10, 12) 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 (3), 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. Informant 11 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 stereotype of the lone computer scientist may be a false one. Informant 4, 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: Send Email or Post News Request for Help.* Neither email nor network news programs were widely used by the informants as a problem-solving resource, although email was used for other purposes by nine out of the 14 informants (see Table III). Only five of the users recalled sending email requests for help, and only one of these (14) preferred email over phone conversations, because it allowed a more detailed communication.

Three of the users who said they would never use email gave time constraints as the reason. "If a problem comes up, you don't want to wait on it," explained Informant 1. The other users simply stated that they didn't use it or that they preferred phone calls.

None of the informants recalled posting to network news with a software problem. Informants 2 and 11 said they would never put a question on network news because it was too humiliating. Informant 2 also thought news was too slow. "I don't want to come back two hours later and get 20 replies." Informant 6's expectations were even lower: "If you post something saying 'does anybody know how to do this?' then you get back 12 replies from people saying, 'yeah, when you find out, tell me.'"

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 (1, 3, 11, 12, 14), 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 four (4, 6, 8, 13) clearly recalled noticing the effects first. 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 1 pointed out that this kind of learning event happened more frequently when his work group had all just started on a new system. Informant 10, who could not recall an example, is a faculty member who explained that he almost never worked in an environment where other people were using computers. Informant 13 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 (3) mentioned user groups as a place to learn things; another (11) suggested videos. Neither recalled specific instances. A third informant (14) 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. It is of interest to compare informants' behavior in the domain of computers to their behavior in other domains. It might be that some users are especially inclined to exploratory activities in general, an inclination that could show up in their computer activities. Conversely, if the users did not explore 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 noncomputer domains. Because the words "exploratory activities" would have little meaning to the informants, the question was supplemented by examples: traveling without a detailed itinerary or shopping with nothing to buy.

All 14 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. All of the exploratory activities described by informants were recreational, which fits with 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.

4. GENERAL DISCUSSION

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 workplace, under the constraints of their daily jobs, with particular attention to strategic use of available resources. The interviews were designed to validate the logs and extend the study's coverage to strategies for initially learning new computing systems. Taken together, the logs and interviews provide converging evidence for several key points:

—Users are primarily concerned with accomplishing the tasks that define their jobs. The time pressures associated with those tasks are a major factor in determining what and how a user chooses to learn about a computer system.

—Although many users will scan tutorials, manuals, and menus to gain an initial overview of their software, they generally prefer a "just-in-time," task-driven approach to learning the details. Exploratory learning for pleasure is rare.

—Trying different things with a system to discover how to accomplish a task is an approach that users apply frequently and successfully in the workplace. Although this approach can be successful on its own, it is often combined with looking in manuals and asking other users or system support personnel for help.

—On-line help is widely disliked, although it is sometimes used in conjunction with trial and error.

These results describe the behavior of a relatively well educated, computer-sophisticated group of users, most of whom have had long-term access to modern computing facilities. 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. This is a situation that may be widespread in the culture of today's workers in large organizations, although it 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. Indeed, it is interesting to compare these results with those reported in Rosson's [1984] survey of 121 users of the Xedit word processing system. Those users reported their most important learning resources as the manual (46.9% of those surveyed), experimentation (21.2%), and assistance from other users (17.7%). These are the same three resources that predominate in the current study, but, at least for the ongoing learning reported by the Eureka slips, experimentation has pushed the manual out of first place.

4.1 Implications for Exploratory Learning and Training

The increased willingness to use trial and error supports the direction of design aids such as the Cognitive Walkthrough [Wharton et al. 1994], which is biased toward the development of "walk-up-and-use" systems. However, this recommendation needs to be tempered by the additional finding that trial and error is typically used in conjunction with manuals and asking for help. In the workplace today, users evidently do not expect a system to be entirely comprehensible without some external help, and they have evolved the skills to identify and acquire that assistance. Designers of software applications need to keep those skills in mind as they balance walk-up-and-use learnability against speed and ease of frequent operations.

The study also shows that users often prefer to postpone their learning until driven to it by real tasks. Informants with this attitude would not "do" supplied tutorials as a means of learning a new application. However, they would skim through the tutorials in order to see what functions the program offered and to acquire an overview of the procedures required. Learning the details would be postponed until a real task demanded it. Other users would work through the tutorials as designed, or with slight changes to incorporate their own data. Both kinds of user might be better served if tutorials were specifically designed to support high-level scanning followed by later visits for detailed information. Many on-line tutorials clearly fail in these respects, because they must be stepped through from beginning to end or because they take over control of the primary application.

The limited role that formal, application-specific training was found to play in the lives of many users suggests the need for training in on-the-job learning strategies themselves. The more experienced informants had well-defined approaches for learning new programs and resolving problems. The just-in-time, task-oriented strategy for learning software was one example. Another was the resolving of problems by alternately trying things out on the system and looking for external help. These general strategies reflect a knowledge of which resources to access for certain kinds of information, along with a repertoire of techniques, such as saving a large file before trying a new global command. Draper [1984] makes a similar point, noting that there are essentially no users with global expertise in Unix, but the more experienced users know how to experiment with the system, where to look for answers to their questions, and who to ask when those resources fail. It could prove valuable to train novice users in these skills.

4.2 Implications for Documentation and Support

Manuals, on-line help, and phone-in help lines are some of the features typically supplied with software to support training and ongoing use. Wright [1988], in her review of documentation design, describes a range of functions that this documentation needs to serve, from tutorials to quick references to detailed explanations. To meet these needs, many PC and Macintosh programs today come with several major pieces of documentation, with titles such as "Getting Started," "Tutorial," "User Manual," and "Reference."

The informants in this study, especially those with wide experience, expressed a familiarity with these resource distinctions. Some of them described how they would use different items to access different kinds of information, such as an initial survey of a package's functionality, an overview of a procedure, or the step-by-step details. This not only supports the need for multiple resources; it points to an opportunity for leveraging off existing user skills. Documentation will benefit from maintaining a consistency with the formats that are commonly found on a given platform.

For task-oriented problem solving, the informants would typically use the documentation in combination with trying things out on the system itself, an approach that is intentionally supported by "minimal" manuals [Carroll 1990]. This strategy also calls for more than one kind of documentation, including global or overview materials that provide a conceptual framework for exploratory activity, and detailed references to consult when trial and error fails [Holt et al. 1989; Lewis and Rieman 1994].

It is of interest that some users in the field study relied on third-party manuals. This may be because these books are more easily acquired by individuals in a site-licensed situation. However, another possibility is that third-party texts help to overcome the "vocabulary problem" noted by Furnas et al. [1987]. The manuals supplied with the system are likely to use the same concepts and vocabulary as the software itself, which may have already failed for the user during trial-and-error exploration. Third-party texts can provide an alternative view of the same material.

The alternative viewpoint of third-party texts may also help to trap and refocus low-level questions that reflect an erroneous mental model of the software. An example of this problem was observed in laboratory trials related to the field study. Franzke and Rieman [1993] gave users an unfamiliar application and asked them to create a graph from a spreadsheet. A few subjects did not realize that the program could automatically plot the data; these users ignored the menus for graph creation and spent their time looking for drawing tools. When this kind of problem occurs, even a moment's assistance from another user may be critical. It is also the kind of problem that intrinsically motivating overviews or tutorial materials may help to avoid, by catching the busy user's attention and highlighting key concepts.

4.3 Implications for On-Line Help

This study did not include a detailed assessment of the on-line help systems available to each informant, but the range of systems was considerable. They included the hypertext-style Symbolics Document Examiner, the help systems supplied with various applications on Macintosh and PC platforms, and the command-oriented "man" and "help" of Unix and VMS. Although there were some instances where individual informants identified a particular help system as useful, the overall finding was that on-line help was inferior to manuals.

This result is paradoxical given the task-oriented and time-constrained nature of users' learning activities. Of all the external learning aids, on-line help is the one resource with immediate access to at least some part of the user's current context. It may also be the most readily available, especially in the increasingly common situation where users of site-licensed software or laptop computers do not have copies of the manuals at hand.

A variety of reasons have been suggested for the weakness of on-line help systems [Duffy et al. 1992]. One simple explanation is that the help window obscures the task window. More sophisticated arguments in the same direction involve cognitive load and interruption [Wright and Lickorish 1994]. The interviews and the Eureka data in Table VII lend support to these ideas. 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.

Additionally, the time constraints that users found so important may seriously discourage them from using on-line help. On-line systems may be slower than printed documents—or may appear slower—because of the reduced readability of on-line text [Gould et al. 1987], the need to wait for disk access or screen repainting, and the need to scroll through text in a small window. The system will be especially discouraging to the user who is visually scanning a large amount of help text in search of detailed information described in an unfamiliar vocabulary. Two on-line systems that actively target the vocabulary problem are the 1-2-3-to-Excel command translation provided by Microsoft, and Superbook, using latent semantic indexing [Egan et al. 1989].

4.4 Implications for Further Research

The results of this study identify the context in which exploratory learning typically takes place. Task-free exploration is unlikely to occur, not only because systems are uninteresting, but because the users' fundamental goal is to complete their work-related tasks. To that end, they will use their knowledge of the available resources to choose the fastest, most focused learning strategies available. 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.

This real-world data from the field, along with broader data from instruments such as surveys or more focused results from on-site videotaping, can be used to improve the contextual validity and relevance of laboratory research. This is a partial answer to the problems identified by proponents of situated cognition and related design methodologies [Suchman 1987; Wixon et al. 1990]. The data have already helped to shape recent research in our laboratory [Franzke 1995; Franzke and Rieman 1993; 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. Jack Carroll, three anonymous *TOCHI* reviewers, Richard Thomas, and my colleagues at the MRC Applied Psychology Unit in Cambridge helped shape the final draft. The 14 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 Comput. 15,* 11, 49–58.

Carroll, J. M. 1990. *The Nurnberg Funnel.* MIT Press, Cambridge, Mass.

Carroll, J. M., Mack, R. L., Lewis, C. H., Grischkowsky, N. L., and Robertson, S. P. 1985. Exploring a word processor. *Hum. Comput. Interact. 1,* 283–307.

Carroll, J. M. and Mazur, S. A. 1986. Lisa learning. *IEEE Comput. 19,* 10, 35–49.

Carroll, J. M. and Rosson, M. B. 1987. The paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction,* J. M. Carroll, Ed. MIT Press/Bradford Books, Cambridge, Mass., 80–111.

Draper, S. W. 1984. The nature of expertise in UNIX. In *Human-Computer Interaction—Interact '84.* Elsevier Science, Amsterdam, 465–471.

Duffy, T., Mehlenacher, B., and Palmer, J. E. 1992. *On Line Help: Design and Evaluation.* Ablex, Norwood, N.J.

Egan, D. E., Remde, J. R., Gomez, L. M., Landauer, T. K., Eberhardt, J., and Lochbaum, C. C. 1989. Formative design-evaluation of Superbook. *ACM Trans. Inf. Syst. 7,* 1, 30–57.

Engelbeck, G. E. 1986. Exceptions to generalizations: Implications for formal models of human-computer interaction. Masters thesis, Dept. of Psychology, Univ. of Colorado, Boulder, Colo. Unpublished.

Ericsson, K. A., Tesch-Römer, C., and Krampe, R. T. 1990. 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, 109–130.

Fischer, G. 1987. Cognitive view of reuse and redesign. *IEEE Softw. 4* (July), 60–72.

Franzke, M. 1995. Turning research into practice: Characteristics of display-based interaction. In *Proceedings of the Conference on Human Factors in Computing Systems.* ACM, New York, 421–428.

Franzke, M. and Rieman, J. 1993. Natural training wheels: Learning and transfer between two versions of a computer application. In *Proceedings of the Vienna Conference on Human Computer Interaction 93*. Springer-Verlag, Berlin, 317–328.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. 1987. The vocabulary problem in human-system communication. *Commun. ACM 30,* 11 (Nov.), 964–971.

GOULD, J. D., ALFARO, L., FINN, R., HAUPT, B., AND MINUTO, A. 1987. Reading from CRT displays can be as fast as reading from paper. *Hum. Factors 26,* 5, 497–515.

HOLT, R. W., BOEHM-DAVIS, D. A., AND SCHULTZ, A. C. 1989. Multilevel structured documentation. *Hum. Factors 31,* 2, 215–228.

JOHN, B. E. AND PACKER, H. 1995. Learning and using the cognitive walkthrough method: A case study approach. In *Proceedings of the Conference on Human Factors in Computing Systems.* ACM, New York, 429–436.

LEWIS, C. 1988. Why and how to learn why: Analysis-based generalizations of procedures. *Cog. Sci. 12,* 211–256.

LEWIS, C. AND RIEMAN, J. 1993. Task-centered user interface design: A practical introduction. 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, Mass.

MALONE, T. W. 1982. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *Proceedings of the Conference on Human Factors in Computing Systems.* ACM, New York, 63–68.

NARDI, B. 1993. *A Small Matter of Programming.* MIT Press, Cambridge, Mass.

NEAL, L. R. 1987. Cognition-sensitive design and user modeling for syntax-directed editors. In *Proceedings of CHI+GI '87 Conference on Human Factors in Computing Systems and Graphics Interfaces.* ACM, New York, 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. In *Proceedings of CHI'86 Conference on Human Factors in Computing Systems.* ACM, New York, 162–167.

POLSON, P. G. AND LEWIS, C. H. 1990. Theory-based design for easily learned interfaces. *Hum. Comput. Interact. 6,* 191–220.

RIEMAN, J. 1993. The diary study: A workplace-oriented tool to guide laboratory studies. In *Proceedings of the InterCHI'93 Conference on Human Factors in Computer Systems.* ACM, New York, 321–326.

RIEMAN, J. 1994. Learning strategies and exploratory behavior of interactive computer users. Ph.D. dissertation, Tech. Rep. CU-CS-723-94, Dept. of Computer Science, Univ. of Colorado, Boulder, Colo.

ROSSON, M. B. 1984. Effects of experience on learning, using, and evaluating a text-editor. *Hum. Factors 26,* 463–475.

SCHNEIDERMAN, B. 1983. Direct manipulation: A step beyond programming languages. *IEEE Comput. 16,* 8, 57–69.

SHRAGER, J. 1985. Instructionless learning: Discovery of the mental model of a complex device. Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, Pa. Unpublished.

SHRAGER, J. AND KLAHR, D. 1986. Instructionless learning about a complex device: The paradigm and observations. *Int. J. Man Mach. Stud. 25,* 153–189.

SUCHMAN, L. A. 1987. *Plans and Situated Actions.* Cambridge University Press, Cambridge, England.

WHARTON, C., RIEMAN, J., LEWIS, C., AND POLSON, P. 1994. The cognitive walkthrough method: A practitioner's guide. In *Usability Inspection Methods,* J. Nielsen and R. L. Mack, Eds. John Wiley and Sons, New York.

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.* ACM, New York, 331–336.

WRIGHT, P. 1988. Issues of content and presentation in document design. In *Handbook of Human-Computer Interaction,* M. Helander, Ed. Elsevier (North-Holland), Amsterdam, 629–652.

WRIGHT, P. AND LICKORISH, A. 1994. Menus and memory load: Navigation strategies in interaction search tasks. *Int. J. Hum. Comput. Stud. 40,* 965–1008.

Received August 1994; accepted September 1995