Software Use in the Workplace: A Study of Efficiency

by

S. Camille Peres

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

Doctorate of Philosophy

APPROVED, THESIS COMMITTEE:

David M. Lane, Associate Professor, Chair
Psychology and Statistics

Michael D. Byrne, Assistant Professor
Psychology

H. Albert Napier, Professor
Management and Psychology

Christopher M. Kelty, Assistant Professor
Anthropology

HOUSTON, TEXAS
APRIL 2005
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 3168123
Copyright 2005 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346
Software Use in the Workplace: A Study of Efficiency

Abstract

by

S. Camille Peres

Although existing laboratory research shows that software is often used inefficiently, relatively little is known about (a) how efficiently software is used in a real work environment and (b) the factors that influence the efficiency of individual users. The present research consists of an ethnographic investigation of software use in an office setting. The study occurred over a four-week period and consisted of observations, in-depth interviews, and an online survey of employees at a large energy company in Houston, Texas. Employees frequently used their software inefficiently and when they were efficient, were not consistent with their efficient use of software. They tended to approach using and learning software programs differently depending on whether the program was associated with their area of expertise. For those programs associated with their expertise, they were more likely to seek out learning new and more efficient methods of using the software for doing their job. However, for other software they would learn as little about it as possible, even if they spent most of their working day on the latter type of software. Furthermore, employees consistently reported that, regardless of the complexity of the program, they primarily learned to use programs by exploring the interface. Through this exploration, they would learn enough about the program (or would reach a sufficient level of knowledge) to do their job. Any knowledge of alternative features or efficient methods of using these features would subsequently be learned through peers.
ACKNOWLEDGMENTS

Of course, I would first like to offer my sincere gratitude to my advisor, David Lane. The last four years have been an extraordinary experience and his counsel and guidance has been invaluable part of that experience. I hope I enjoy my next position as much as I have enjoyed working with him. I would also like to thank the other members of my committee, Michael Byrne, Al Napier, and Chris Kelty. I will forever refer to them as the “Dream Team.” Each member of this committee provided important and unique insights to this project and thus the results are fuller and richer because of their guidance. Further thanks go to the employees of the Information Technology Corporation within ChevronTexaco and in particular Ken Knowles, without whose efforts this project would not have been possible. Not only did he unselfishly give of his time and expertise, he made sure I ate lunch every once in a while as well. More gratitude goes to my colleagues Anikó Sándor and Minmin Yang. Their moral support over the last year has been wonderful!

I would also like to recognize my friends and family. When I think of the network of people who have supported and encouraged me throughout the years, I am overwhelmed. In the most literal sense, this would not have been possible without all of them. From my neighbors who would meet my children at the bus if I were running late, to my parents who kept the children over night so I could stay in town and work. To all of you—my friends and neighbors; Dad, Linda and Kendra; Mom and Herman; and my children, Landon and Audrey—thank you, thank you, and thank you.

Finally, and most importantly, to my husband—you were right, I admit it. I could do all of this, but not without you there to support and encourage me. Thank you.
# Table of Contents

Abstract................................................................................................................................. iii

Acknowledgments.................................................................................................................. iv

Table of Contents.................................................................................................................... v

List of Tables ........................................................................................................................... vi

Software Use in the Workplace: A Study of Efficiency ......................................................... 1

Ethnographic Methodology .................................................................................................. 32

  Current Study ....................................................................................................................... 36

Results.................................................................................................................................... 41

  Field Notes .......................................................................................................................... 42

  Efficiency with software ..................................................................................................... 70

  Observations on learning software ...................................................................................... 83

Discussion............................................................................................................................... 89

  Efficiency............................................................................................................................. 89

  Learning ............................................................................................................................... 93

  Other observations ............................................................................................................. 97

Conclusions ............................................................................................................................. 98

References............................................................................................................................... 101

Appendices.............................................................................................................................. 109
LIST OF TABLES

Table 1. Frequencies of programs used by respondents as a function of the group and the type of software. .................................................................................................................. 76

Table 2. Frequencies for: "How many times in the last 6 months have you learned something new about ____?" ........................................................................................................ 83

Table 3. Frequencies for: "The last time you learned something new about ____, how did you learn it?" .................................................................................................................. 83

Table 4. Number of times respondents ranked 1st the type of information they wanted from the Help Desk. ........................................................................................................ 87

Table 5. Number of times respondents ranked 1st the type of information they received from the Help Desk. ........................................................................................................ 88
SOFTWARE USE IN THE WORKPLACE: A STUDY OF EFFICIENCY

The proliferation of personal computers has brought powerful tools to the desktop both at home and work. These powerful tools often consist of software that allows users to issue hundreds of commands. People use this software for tasks ranging from complicated ones such as creating architectural blueprints to simple tasks such as printing a letter. These programs are used not only for a variety of tasks but they also are often used for a large portion of someone’s working day. Studies have shown (Davis, Smith, & Marsden, 2002; Lane, Napier, Peres, & Sándor, 2005; Peres, Fleetwood, Tamborello II, Yang, & Paige-Smith, in press) that over 50% of the people who use a computer at work report using the computer over 10 hours a week. The software programs people use on the job are often integral to their work.

Licensing and training costs associated with software are very expensive. Businesses spend time and money on software assuming its use will improve communication, support better decision-making, and ultimately increase productivity and profit. In order to realize these benefits, it is important that the time employees spend learning software not be excessive. The longer training takes, the longer the employee is away from his or her normal tasks and the longer it is until they can realize the benefits of the new software. Furthermore, in order to get the maximum value out of the dollars spent on the software and the training, employees should become knowledgeable about the program’s features and how to best use these features for their work.

Productivity benefits based on the adoption of computer technology have been hard to demonstrate. In his book *The Trouble with Computers* Landauer (1996) argues that the capital and training investments made in information technology by companies
since the 1960s have not resulted in a corresponding increase in productivity. In an attempt to unravel this “productivity puzzle,” business executives as well as human computer interaction (HCI) professionals and researchers have proposed several explanations. Landauer places the blame at the foot of the designers of the technology itself. He suggests that interfaces are so poorly designed that users cannot learn them easily or use them efficiently. He argues that until the needs and capacities of the user are fully integrated into the design process (user centered design) the technological tools that result, regardless of the power or ingenuity of the tools, will not result in the desired increase in productivity. It should be acknowledged, however, that other points of view exist. For example, Brynjolfsson and Yang (1996) argued that much of the “missing” productivity gains are due to poor measurement methods. Nonetheless, Landauer’s point that the productivity gain is far less than it could be is well taken.

Overview of Current Research

One aim of the current research is to gain a better understanding of the “productivity puzzle,” particularly as it relates to the efficiency with which people use computers and the factors associated with these levels of efficiency. This discussion will begin with a review of what is known about users’ efficiency levels with computers. Then, an examination of what constitutes efficient use of software will be presented along with what can facilitate or impede someone’s progression to becoming an efficient user of software. The research presented here consists of an ethnographic investigation of the efficiency of software use and addresses several issues including (a) the extent to which software is being used efficiently, (b) intra-individual differences in efficiency, and (c) how people learn to become efficient users.
Early Interfaces

Most of the interfaces in the 1970s and early 1980s were command-line interfaces. With these interfaces, the challenge of learning and developing expertise was not trivial. Being “afraid” of computers was a realistic fear as it was possible with a few misplaced keystrokes to delete all of one’s work or wipe out an entire program. In fact, some of the early research in HCI documented how difficult it was for users to learn these interfaces (Carroll & Rosson, 1987; Mack, Lewis, & Carroll, 1983; Rosson, 1983).

The advent of the Graphical User Interface (GUI) resulted in important progress regarding the learnability of software. Of the many benefits of the GUI, the more notable are: (a) the inclusion of direct manipulation, allowing the user to directly manipulate objects on the screen with a mouse, (b) design conventions, allowing the transfer of knowledge from one program to another, and (c) having many of the commands visible while using the software, thus eliminating the need to memorize commands. Indeed, it is now widely accepted that interfaces presenting information “on the device” are easier to use than those that require knowledge “in the head,” due to the decreased demands on memory (Zhang & Norman, 1994). With the GUI, people are able to walk up and use many programs without needing training or tutorials (Galitz, 1996). Thus, the learnability of most programs is dramatically higher with GUIs than with a command-line interface.

Dark side of the GUI

Despite these improvements to the interface brought about by GUIs, users generally learn only a small proportion of the features in the programs they use (Linton, Joy, Schaefer, & Charron, 2000). Therefore, although the GUI has dramatically improved people’s ability to learn enough about a program to do their jobs, it may not be adequate
to ensure the user’s transition from a novice to expert performance level. Because the GUI makes many of the possible commands visible while someone is using the software, these interfaces can result in users only selecting the best option of those visible and not considering the adoption of more effective and/or efficient methods (Polson & Lewis, 1990). Thus, people may learn just enough to accomplish their work, and not necessarily explore the software further to learn additional features available to them. Carroll and Rosson (Carroll & Rosson, 1987) foresaw this possibility in their discussion of the active user of software and warned that

‘... there remains the possibility of a more general cost of eliminating the assimilative gap; if learners are no longer required to ‘work’ for the new knowledge, they may fail to engage in the active processing critical to building a rich and flexible understanding’ (Carroll & Rosson, 1987, p.98).

People can be efficient

Despite Carroll and Rosson’s warning, laboratory research has revealed examples of efficient software use. In a series of experiments, Charman and Howes found that people will use efficient methods (Charman & Howes, 2001, 2002, 2003) and one experiment in particular investigated the conditions that would support the adoption of efficient methods (Charman & Howes, 2001). Users participated in training on basic drawing skills in PowerPoint that included exposure to all of the functions they would need to use to perform the experimental task efficiently. For all of the tasks, participants were instructed to complete the drawing tasks in as few moves as possible. They found
that although most people would initially use the less efficient method of working with one shape at a time, they would eventually use the efficient method, including the method of selecting more than one item at a time and then duplicating and manipulating the items they had selected. Although this suggests that people can discover efficient method with practice, it is conceivable that these effects were due to the fact that participants were primed for efficiency as they were expressly told to complete the task in as few moves as possible. It is therefore not known how well this finding would generalize to the work environment.

Jorgensen and his colleagues (2002) found that after extended practice, users preferred a more efficient method to a less efficient method. Participants in their experiment performed a speeded classification task (i.e., Stroop task) for two hours. For one of those hours they responded using the mouse and for the other hour, they responded using the keyboard (order of response method was randomized). Not surprisingly, they found that participants performed the task much faster with the keyboard than the mouse. However, of more interest was the participants’ preference—they overwhelming preferred using the keyboard by the end of the experiment. Many of them mentioned that they had expected to prefer the mouse, but found it to be slow and uncomfortable enough during the course of the experiment that they ultimately preferred the sessions where they used the keyboard. It is unclear, however, whether users would have ever discovered that the keyboard was more efficient than the mouse if they had not been required to do the task with the keyboard.
Efficiency with software

Although these results provide some evidence that users, on occasion, are efficient, they do not contradict the notion that users are generally inefficient. Moreover, the results may not generalize to a working environment where multiple factors affect how someone uses software.

Levels of efficiency—Past. Although there is not much information available about current levels of efficiency in the workplace, there is information about the patterns of behavior seen in the past. Carroll and Rosson (Carroll & Rosson, 1987; Rosson, 1983) did a series of studies examining the use of text editors and found that users did not reach the expert level of performance, regardless of the amount of time they had spent using the software. This finding led naturally to the question, “Why do people maintain sub-optimal level of performance with computer programs?”

After observing people learning and using the text editor programs, Carroll and Rosson presented their account of why users have difficulty learning new software and why they tend to asymptote at moderate (and even low) levels of performance (Carroll & Rosson, 1987). They identified two factors: the production bias and the assimilation bias. The production bias stems from the fact that the user’s paramount goal is to complete the task as accurately and quickly as possible. Although this is often an appropriate goal, it conflicts with the goal of learning the features of the software or the system. This conflict between needing to stop working on the task to learn something new and not wanting to stop working on the task in order to get the job done establishes a “motivational paradox” within the user. With the production bias, people will stick with procedures they already
know in order to finish the task and receive reinforcement for that task. They illustrate the effects of this paradox in the quote below:

"Designers of reference and help systems count on users to recognize opportunities for new methods, and to search out the information needed to implement them. But users often figure out how to use what they already know to achieve new goals. They have little desire to explore new functions or to search out information, if they can use methods they are already comfortable with to achieve the same goal."

(Carroll & Rosson, 1987, p.82)

In contrast to the production bias, which explains why users typically fail to reach an expert level of performance, the assimilation bias explains why users often have trouble learning to use software in the first place. The assimilation bias is the user's tendency to apply their existing knowledge of a task when learning the same task using software. Although assimilation can facilitate the learning process, it makes the learning process difficult when users misapply their current knowledge of the task. This creates the "cognitive paradox" which states that someone who has existing knowledge of the task may have more trouble learning how to do that task on a computer than someone who has no knowledge of the task at all.

The text editors on which Carroll and Rosson based their observations differ dramatically from those used today, and it is conceivable that some of the same observations do not generalize to someone using a text editor (word processor) in 2005. It is also possible that their findings were more a function of people not only having to learn
software, but also having to learn how to use computers in general. Nevertheless, their explanation of users’ difficulties with learning new software and asymptoting at sub-optimal levels of performance provided a relevant framework for investigating this phenomenon.

Bhavnani and his colleagues conducted a more recent study of efficiency with an ethnographic study of computer aiding drafting (CAD) software usage (Bhavnani et al., 1996). They found that architects were using the CAD software extremely inefficiently. Bhavnani observed both the cognitive and motivational paradoxes with architects at work. The users communicated that their main focus was to “get the job done” (production bias) and although they were motivated to work as efficiently as possible, most of them simply did not know about the features of the program and did not want to take the time to learn them (motivation paradox). They observed that users would often apply methods and strategies for doing a drawing by hand when they were doing a drawing with CAD (assimilation bias). Often these strategies were efficient when making a drawing by hand, but with the CAD, they may be not only inefficient, but also ineffective (cognitive paradox).

They also found that those users who had learned new commands did so through casual conversation or chance observation. Furthermore, some of the users stated that they would be more likely to use a new command if a colleague had used it because he or she would know that it would be useful to them. Although this study collected important information regarding the failure to use efficient method, the information is limited to the use of CAD software and was conducted almost 10 years ago. Even in this short period,
the use of computers in home and office situations has increased dramatically, thus these results may not be generalizable to a current office setting.

To date, the research regarding the efficiency with which people use software is incomplete. There is strong evidence that when people began using text editors and CAD software, their use of these programs did not progress beyond a level just sufficient to accomplish their tasks (Bhavnani et al., 1996; Carroll & Rosson, 1987). Although there is evidence that under certain circumstances users will discover and prefer efficient methods (Charman & Howes, 2001; Jorgensen et al., 2002), it is not clear that these behaviors are being seen in the workplace. Indeed, self-report data suggest that just the opposite is occurring (Lane et al., 2005; Peres et al., in press), that is, despite the improved learnability of the GUI people may be just as likely now to be inefficient as they were in the mid-1970s.

Levels of efficiency—Current. Although improvements in the design and interface of much of the software people use on a daily bases would lead many to expect an improvement in the efficient use of software, anecdotal evidence indicates otherwise. However, there is currently little research documenting how efficiently (or inefficiently) people are using software at work, and much of what is known suggests they are not very efficient. For instance, two studies investigating the use of the keyboard to issue commands in Microsoft Word found that most experienced users do not employ this efficient method even though it takes approximately one-third the time as using the mouse and is relatively easy to do (Lane et al., 2005; Peres et al., in press).

One of these studies (Lane et al., 2005) asked professionals to indicate the percent of the time they used different methods to issue 14 different commands and found that
fewer than 20% of the respondents used the keyboard to issue even the most frequently used commands (e.g., Cut, Copy and Paste). In addition to the low usage of this more efficient method, Lane et al. found no relationship between the use of the keyboard to issue commands with age, computer experience, or time spent using Word. In a similar study with undergraduate students (Peres et al., in press), the average use of the keyboard to issue commands was only 30%. These results illustrate that even a group of younger, more computer-savvy individuals, did not employ efficient methods frequently.

Although these studies are valid indicators of whether people use software efficiently, they are very narrowly focused and based on self-report. To get a complete picture of how efficiently software is being used in the workplace, it is necessary to go into the workplace and see software use in action.

*Intra-individual consistency.* One explanation for the finding that there are people using software inefficiently is that some people place a higher priority on efficiency that others. It could be that the investigation of the level of efficiency for the “general user” is misguided and that there are those who consistently learn and use efficient methods of doing things and those who do not. Many people can easily think of a few people considered computer “geeks” or “tinkerers” who will learn as much as they can about almost every software program they know. These people seem to have a general tendency to seek out and adopt the more efficient methods of using software.

Indeed the theory of goal orientation, a theory developed with regard to children’s performance in education (Dweck, 1986, 1991) and now widely applied to organizational behavior, states that when learning new concepts or material, a person can adopt either a learning orientation or a performance orientation. People with a learning goal orientation
seek to increase their competence by understanding or mastering a new topic. Those with a performance goal orientation are more concerned with receiving favorable judgment of their performance or at least to avoid negative judgments of their competence. Studies have found that goal orientation differentially affects performance in school for children (Dweck, 1986; Eppler & Harju, 1997; Ford, Smith, Weissbein, Gully, & Salas, 1998; Schmidt & Ford, 2003) and in training settings for adults (Colquitt & Simmering, 1998; Ford et al., 1998; Schmidt & Ford, 2003).

According to goal orientation theory, users with a learning orientation would be motivated to become efficient with software: They would investigate the features of the program, do the tutorials, and possible even read the manual. In contrast, those with a performance orientation would not put much effort toward learning efficient methods since they would be able to produce quality products without those methods, albeit inefficiently. Since being efficient is often not an overt performance objective in work settings, someone with a goal orientation focused on receiving favorable judgments of their performance would not see efficiency as a priority. This perspective could also explain the use of efficient strategies in the laboratory (Charman & Howes, 2001, 2002). In these studies, participants were told that performing the task in as few moves as possible was the goal and thus a measure of performance. For this situation, both the learning and performance orientation perspectives would be motivated to utilize efficient methods, although the source of their motivation would be different. For those with a learning orientation, they would use efficient methods because they wanted to become proficient with the tool and for those with a performance orientation, they would use efficient methods because it was a measure of their performance.
The road to efficiency

Regardless of someone's level of efficiency or motivation for acquiring that level, it is important to consider the elements of being an efficient user of software. What do these people know and how is their use of software different from someone who is not efficient?

What it is to be an efficient user. Efficiency can be formally defined as "the ability to do something well or achieve a desired result without wasted energy or effort, or the degree to which this ability is used" (Encarta®, 1999). This suggests that when someone is efficient, he or she performs well and without wasted effort. To translate this to software use, it would mean that a person would be able to successfully and accurately complete a task in the least amount of time and/or with the smallest number of mental operations, mouse clicks, keystrokes, etc.

To accomplish this level of performance, a user would need to know the features available in a particular program, know how those features relate to the actual task, and know how to combine the use of the features in a way that minimizes the time and effort for accomplishing the task. To illustrate the differences between each of these areas of knowledge, the development of an imaginary report will be used as an example. The report is to be created using the spreadsheet program Microsoft Excel and must summarize the employee vacation time spent over the last quarter by country, division, and group for the 5,000+ employees all over the world. The summaries are calculated using a statistical package, but must be put into a formal (and formatted) report in Excel. This report is created every quarter must be delivered to the chairperson one week after the data are made available. Although this is a fictitious example, the process of making
the report is based on an example experienced by the author. Furthermore, the author or one of her colleagues has observed all of the inefficient and efficient uses of the software described in the example (personal communications, Christopher P. Peres).

- **Know what features (commands) are available**

  The first level of knowledge that an efficient user must have is knowledge of the features or commands available in the software. To create this type of report in Excel, the user would need to know how to get the data into Excel, how to manipulate the data once they are in Excel, how to format the data appropriately, and how to duplicate this effort every quarter. Space does not permit a listing of all the features in Excel available for these parameters, but a few will be reviewed to provide clear examples of how an efficient user could utilize this powerful software for this report. Furthermore, the user needs to know not only what features or commands are available, but also, the different (and more efficient methods) of issuing those commands.

  It is clear from the description of the report that the resulting table will be very large and it would not be efficient to retype every cell of the table into Excel. Thus, in the example, the efficient user would need to know the copy and paste features of both programs and that the table from the statistical package can be copied and pasted directly into Excel. The (extremely) inefficient user who typed the numbers in by hand would be able to accomplish the task of having the information from the statistical package transferred to Excel, but the time it would take to do this would be much higher than using copy and paste as would the potential for data entry errors.

  The efficient user would also know that the fastest way to issue the copy and paste commands would be through the keyboard. Specifically he or she would use the
arrow + shift keys to select the table in the statistical programs, press Ctrl + C to copy the material to the clipboard, press Alt + Tab to move to Excel, and then press Ctrl + V to paste the material into Excel. Each time the efficient user employed the keyboard to issue a command he or she would spend 1/3 less time issuing that command than by using the mouse. The inefficient user would issue every command through the drop down menus at the top of the screen adding unnecessary time to complete the task and placing unnecessary strain on his or her hands.

- Know how to use features

The next level of efficient use of a software program is having knowledge not only of the features available in the software that are relevant to the task, but also a knowledge of how those features apply to the task. For instance, if the report required the calculation of the percent change from the previous quarter’s data, an efficient user would not only know that formulas were available in Excel but also that they could be used in this situation to calculate this value. In the formula that calculates the percent change from the previous quarter to the current quarter, the efficient user would reference the necessary value in the previous quarter’s spreadsheet. This illustrates not only a knowledge of the fact that Excel has formulas, but also an understanding that through the use of formulas, much of the information in the Excel worksheet can be referenced for calculation. In contrast, the inefficient user may accomplish this same task by printing out last quarter’s report, and using that information, calculate the values on a calculator and then enter the resulting values on the spreadsheet by hand.
• *Know how to combine them (commands/features) efficiently*

Although knowing the features of a program and how those features relate to the task are necessary for efficient use of software, the hallmark of an efficient user is his or her ability to combine the use of features or commands in a way that truly maximizes the effort expended. When considering that the report must be generated every three months and that it must be built quickly, an efficient user might create one spreadsheet that is a template report with all of the appropriate formatting and extra calculations necessary. He or she would have the cells for this report populated by another spreadsheet in which the data were pasted directly from the statistical package. Since he or she would have to spend time linking the cells from the template spreadsheet to the data spreadsheet, it would initially take more time to create this template than to create a single report. In fact, the inefficient user who was hand entering the data and then formatting that table may be done with his or her report before the template is finish. However, with subsequent reports, the time savings would be tremendous because all that would need to be done is to make a new copy of the template file, run the analysis, and then paste the analysis to the appropriate data spreadsheet in the new file.

The efficient user might take this one step further and insure that others could run the report without instructions from the person who created the report. This could be easily accomplished by inserting comments into the template file and indicate through highlighting and formatting how to paste the necessary data onto the data spreadsheet.

• *Challenges for the efficient user*

One of the challenges for all users is knowing why some methods are more efficient than others and evaluating the long term and short term gains and losses of a
particular method. In the report example, there were times when the efficient user focused on reducing the immediate time necessary to complete the task. However, there was also a time when the efficient user spent more time than the inefficient user on the task. This was because the efficient user knew that the extra time spent would result in a total time reduction on the task in the long term. Improved performance with software can have seemingly subtle impacts, such as decreasing the amount of time necessary to issue a command, or more profound impacts, like decreasing the likelihood of a Repetitive Strain Injury (RSI) (Aronsson, Dallner, & Åborg, 1994). Indeed, Jørgensen et al. (2002) found that participants reported less musculoskeletal discomfort using the more efficient method of issuing commands with the keyboard rather than mouse. An efficient user has gained knowledge and information regarding both the subtle and not so subtle benefits of using the different methods available in the software.

As presented, the timesaving benefit of creating the template was profound; however, it is often that the differences in efficiency between two methods are subtler. For instance, the total cost of not using the keyboard to issue commands may not be large for any one command, however when the total cost in time and in unnecessary strain on the hands is taken into account, the efficiency of using this method to issue commands seems much larger.

Factors that affect the development of efficiency

- Selection Rules

Efficient use of software entails using an efficient method rather than an inefficient method. To understand why someone would or would not use an efficient method, it is first important to understand the cognitive processes associated with this
behavior. In 1983, Card, Moran and Newell (1983) outlined four components of the
cognitive structures associated with task performance. Those components are: (1) the
user’s Goals, (2) Operators, (3) Methods for achieving the goals, and (4) Selection rules
for choosing which Method to employ. This is the GOMS model of human information
processing and provides a helpful framework for discussing when and why someone
would select an efficient method versus an inefficient method of performing a task.
Specifically, when someone has a set of methods for performing a task, what is the
selection rule that determines which of these methods he or she will use?

When deciding which method they will use, users are often selecting between a
well-known method and a less-known method. For example, when deciding whether to
format a Word document using the Styles feature, some people may know that this
feature exists but they may not know it as well as they know the individual formatting
tools. Thus, they will choose to format each heading and subheading individually as
opposed to taking the time to learn the Styles feature and become as proficient with it as
they are with other formatting tools. Although the predominant reason people choose the
less efficient method of two methods they know is most likely that they are more
proficient or comfortable with the less efficient methods, some research has suggested
that proficiency is not the full explanation.

As might be expected, users often prefer efficient methods (Jorgensen et al., 2002;
Young & MacLean, 1988). Indeed, Young and MacLean (1988) found that participants
preferred the more efficient method of using the keyboard to enter numbers into a
spreadsheet. In this study, participants entered numbers into spreadsheet using either the
mouse or the keyboard. During the initial trials, users were required to use the mouse
method for one block, the keyboard method for another block, and either method for the last block. The mouse method required moving between the mouse and the keyboard and although the keyboard method required a longer set up time initially, it allowed the user to keep his or her hands on the keyboard. Interestingly, participants preferred the keyboard method even during trials when the mouse method was up to 10 seconds faster. Young and MacLean suggest that when people have mastered a particular schema and they know the requirements, parameters, and dependencies of that particular task, they are more likely to weigh some elements of the schema differently than others are. In this spreadsheet instance, they suggest that after the participants had mastered the task, when choosing between the mouse and the keyboard method, they weighed the set up time required for the keyboard method less than the total time for the mouse because of the cost of having to move between the mouse and the keyboard with the mouse method. With this “simple compensation schema,” Young and MacLean were able to predict which method users would choose even when they did not choose the fastest method. This suggests that people may choose a method because it feels more efficient even if it does not necessarily take less time. It may be that the timesavings gained by using the efficient method is outweighed by the cognitive or physical effort required to use the more efficient method.

Fu and Gray (2004) found that participants sometimes used a method that was less efficient than another method they knew equally well. They present these findings through the examination of three situations—one in which students learned how to use a program to create interactive objects (i.e., HyperCard™), another where architects were observed using CAD software (i.e., Bhavnani’s ethnographic study), and a third
controlled experiment designed to investigate their hypothesis. In the first situation, they found that some students (3 out of 9) used an inefficient method to create an object in HyperCard™ even though they had used the more efficient method previously. When examining the methods used by an architect, they documented that the architect used less efficient methods for creating a particular type of drawing even though he had used a more efficient method on other drawings. Thus, even though the architect knew and was comfortable using the efficient method, he still chose the less efficient one. For the controlled experiment, participants had the task of programming a simulated VCR to record a show. When selecting the channel that the show would be recorded from, participants could use “Up” and “Down” buttons, which would increase and decrease the value in the channel setting respectively or a “Jump” button which would increase the value in the channel setting by 32. The default value of the channel was zero, thus if the channel to be recorded was a lower number, it was more efficient to use the “Up” button. If it were a higher number, it was more efficient to use the “Down” button (this would move the channel to the highest possible value of 63 and then continue decreasing the value after that). If the value were close to 32, the most efficient method was to use the “Jump” button and then the “Up” or “Down” buttons. Participants primarily used the most efficient method of setting the channel value, but interestingly, for the higher values, some participants (6 out of 16) used the “Jump” button even when the “Down” button was clearly more efficient.

Although Fu and Gray presented evidence that people use inefficient methods even when they know the efficient methods, the evidence presented within these three studies is not strong. For the experiment in which students were learning HyperCard™,
although some students did use inefficient methods, overall the students used the more efficient methods the vast majority of the time (out of 292 opportunities, they used the efficient method 271 times). An interesting caveat to the observations with the architect was that the less efficient method he chose when using the CAD would have been the more efficient method if he were doing the same task by hand. Thus, for making the drawings by hand, the architect had an efficient strategy that competed with the efficient strategy for making the same drawing with the CAD. Finally, the results of the simulated VCR programming experiment are certainly equivocal because most users chose the more efficient methods except in those situations where the difference between the most efficient method and the other two methods was very small (e.g., 1 second). Thus, users’ choice in this situation may in fact reflect their difficulty with perceiving the time difference between the three available options.

Fu and Gray concluded that when presented with two methods they know equally well, people use methods that are either general, applying across several programs/tasks, or are interactive, providing immediate feedback on the screen. To support this position, they examined the inefficient methods chosen by the participants in all three studies and were able to illustrate that this was the case. In all three studies, if someone selected an inefficient method, this method was either a general method or an interactive one. However, in all three studies, people chose the efficient method most of the time, especially if it were clearly the more efficient method. Thus, these findings may provide some insight into why people choose an inefficient method when they know an efficient one equally well, but they do not provide much information regarding why people do not progress to an expert level of performance.
Training

In response to the proliferation of software and the need for people to learn how to use it, an entire industry has developed with the goal of teaching people how to use software. People can learn through the traditional classroom-training environment, with self-study manuals, by using advanced organizers, through watching videos, or by searching online.

Regardless of the training method, research suggests that learning how to use software has unique challenges not currently addressed by training curricula. For instance, given the production bias described by Carroll and Rosson (1987), users are often ready to “get to work” and prefer training methods that will allow them to do this. Support for this was found in an investigation of the effectiveness of two types of self-paced study materials designed to teach a word processor (Carroll, Mack, Lewis, Grischkowsky, & Robertson, 1985). One type of study material was the more common self-study manual that contained instructions and exercises for the “learners” to complete. It was structured in a step-by-step manner and learners were expected to move through the steps in sequence. The other type of study material was designed to facilitate “guided exploration” of the software. Learners had a set of cards and each card had concise information regarding a particular task (e.g., typing something). Each card contained information on how to do the task, what errors to watch out for, and how to recover from any errors the user (learner) might make. Although the subjective reaction of the users to the two different types of learning materials were mixed, the performance results were very clear. Users with the guided exploration cards were able to create a better product in a shorter period than those using the traditional self-study manual method. The authors of
this study argued that users benefited from being able to work on a task that they had
initiated as opposed to an artificial exercise given by the manual. Furthermore, they
suggest that merely reducing the amount of material the user needs to read and process
may facilitate learning.

In a field study, Simon and Werner (1996) compared the effectiveness of three
different training methods. They trained 120 individuals on a data processing program.
Participants were assigned to one of three training groups: a self-paced exploratory
group, an instructional group that received traditional classroom training, and a behavior-
modeling group. The behavior-modeling training was similar to an “on the job” training
program in which the participant learned the program while working with someone who
already knew the program. The behavior-modeling group outperformed the other two
groups on all measures and most dramatically on the measures of procedural knowledge.
Simon and Werner suggested that when learning new software, trainees benefit from
being able to practice the task and immediately get feedback on their performance as
opposed to the self-paced exploratory group who got no feedback or the classroom-
training group who were not able to use the program while they were learning it.

There are many methods for learning software and some of the more promising of
these methods involve allowing the user to learn how to use the software while they are
working on a task. These methods have proven more effective at teaching people what
the features of the program are and how to use them than those methods requiring the
user to take a more passive role (e.g., classroom instruction). Nevertheless, there is little
to suggest that these methods take the user to the next step of learning how to use the
features efficiently. Thus, although some of the training methods currently available may
be better than others at getting people to a level of performance sufficient for doing their jobs, these training methods still do not help the user progress to an efficient level of performance.

- **GUI**

  As discussed previously, the interface itself can be a learning tool for the user. When learning and using software, people can explore functions not previously known to them. One way to encourage users to learn become more efficient is to design an interface that explicitly assists the progression to an efficient level of performance. Several different attempts have been made to do this, but most of the results have been discouraging. For instance, efforts to teach users about keyboard shortcuts have included showing the keyboard shortcuts next to the command name in the drop down menu and having the letters underlined for keyboard-activated commands. These methods have met with limited success as evidenced by the fact that most people do not issue the commands with the keyboard, but instead use the more inefficient method of moving the mouse to the menu or icon bar.

  Others have attempted to employ a training-wheels approach to using software in which the initial interface the user operates in is designed specifically to reduce the cognitive load on the learner (Kalyuga, Chandler, & Sweller, 1998) and reduce the likelihood of errors (Carroll, 1983). Research conducted using this type of interface has been promising (Carroll & Carrithers, 1984), however this work has primarily investigated how well someone initially learned to use the software and has not necessarily focused on increasing expertise or improving efficiency. Another important issue associated with this type of training paradigm involves identifying when the full
functionality of the program should be provided to the user. It is not clear whether the user should “know” when they are ready to begin using the full functionality of the program or if program itself should determine this. Given that most users are more interested in “getting the job done” than spending time and energy addressing training, it is conceivable that even if this functionality were available in a program, users would not use it.

- **Intelligent help**

Some promising methods have been presented in the area of intelligent help. Cornett found that when learning how to use the styles function in Microsoft Word, users preferred and performed better with a help system that made suggestions of more efficient methods of issuing commands (Cornett, 1997). However, when Microsoft attempted to offer alternative methods of performing tasks through its online help center “Clippy,” this help was not well received. Clippy tries to teach users when and how to use alternative features in an attempt to provide users with more information about the program, however users often find Clippy annoying and simply turn it off. However, it is not clear whether users turn Clippy off because the advice given is unsolicited, because the help is not “smart” enough to provide the correct information at the correct time, or some other unidentified issue.

Linton and his colleagues have developed and investigated another intelligent help system (Linton, Joy, & Schaefer, 1999; Linton et al., 2000). These researchers developed a recommender system designed to provide information to individual users regarding the functionality of a program they need to learn. The recommender system develops user profiles based on the functions each person is using within a particular
software program. The program compares this profile to the pooled knowledge of his or her peers and then makes recommendations regarding what functions would be the most valuable for the user to learn. Because usage is recorded for employees across a company in order to share alternative methods of performing tasks, it is referred to as OWL – Organization Wide Learning. The premise of the system is promising but its effectiveness at increasing efficiency or the knowledge of features has not been tested. Furthermore, OWL’s method of delivering suggestions to the user is through e-mail. Specifically, on a regular basis (e.g., weekly) users will get an e-mail listing the features being used by other employees in the company with usage profiles similar to theirs. Although the model of providing each employee access to the knowledge and usage of other employees seems to be a good one, it remains to be seen if having the information delivered via e-mail will facilitate the adoption of these methods. It may be more effective to have the suggestions provided while the person is working with the program. If a user were prompted to try another method of performing a particular task while he or she was working, this type of interaction would then be a closer approximation of the behavior modeling approach for learning, and thus may be more effective. Specifically, users would have the benefit of learning while using the software as well as getting feedback (albeit indirect) on their performance.

• **Weight of Pros relative to Cons**

When describing the active user, Carroll and Rosson (1987) discussed some of the motivational challenges encountered by the user, i.e., he or she may want to focus on getting the job done and thus not see the benefit of taking time away from the task to learn new methods for doing the task. Some recent work may shed additional light on
how motivation affects behavior in this environment (Peres et al., in press). This work suggests that, when calculating the costs and benefits of adopting an alternative method, people may weigh these costs and benefits unequally. This study specifically investigated the use of the keyboard to issue commands and found that the weight people gave the benefits of using the keyboard to issue commands was related to the use of the keyboard to issue commands whereas the weight given to the costs of using the keyboard to issue commands was not related. Although these findings are preliminary and are based on self-reported behavior, they nevertheless speak to the issue that users’ motivation to employ more efficient methods may be affected by the perceived benefit of using the efficient methods. Furthermore, they are consistent with the findings of Young and MacLean (1988) suggesting that users’ decisions regarding which method to use are not exclusively based on the actual time to use a method, but with a predictable combination of other factors as well.

- *Teach efficient strategies specifically*

To be efficient, users must learn how and when to use features in addition to learning the features themselves. Lee and Barnard found that users only learned to use software efficiently if the relationship between the features of the software and the task or problem was specifically articulated (Lee & Barnard, 1993). Similarly, Bhavnani suggested that people are inefficient with software because they do not know how to integrate efficient methods of completing a task into their current knowledge of the program (Bhavnani, 2000; Bhavnani et al., 1996; Bhavnani & John, 1996; Bhavnani & John, 1997, 1998; Bhavnani & John, 2000; Bhavnani, Reif, & John, 2001). For instance, they may know that objects can be changed and that more than one object can be selected
at a time, but they may not understand that it is possible to select more than one object and then change all of those objects at the same time. In the series of studies on efficient use of CAD software, Bhavnani and his colleagues discussed the missing link between knowledge of the features in a program and knowledge of how to use them. Specifically, users need specific and overt instruction on how to efficiently combine the features available in a program. To this end, they developed a course that explicitly teaches undergraduate students how to employ more efficient methods of using software programs. The course has been taught at two institutions and has been successful at increasing individuals' use of efficient methods across programs, specifically, UNIX, Microsoft Word, and Microsoft Excel (Bhavnani et al., 2001; Thomas & Foster, 2001).

Although the courses developed by Bhavnani have proven effective at increasing the use of efficient methods, many (if not most) people do not have the opportunity to attend a semester long course to learn efficient methods. Additionally, it is not known what will happen when these students leave the classroom environment. Will they continue to use the efficient methods they have just learned?

• *Peer-to-Peer Learning*

In a study investigating expertise levels on the World Wide Web, Chadwick-Dias, Tedesco, and Tullis (2004) found that those who had more opportunity to learn about the Web in a collaborative learning environment (i.e., where they were around other people using similar software) were more likely to have a high expertise level than those with little opportunity for collaborative learning. Specifically, performance was not related to how much experience someone had on the Web (regardless of their age), but was related to how much opportunity people had for collaborative learning. Evidence for learning
from peers has been obtained in several other studies (Bhavnani et al., 1996; Brown & Duguid, 1991; Carroll, 1997; Carroll & Rosson, 1987; Simon & Werner, 1996) including some recent work (Peres, Tamborello II, Fleetwood, Chung, & Paige-Smith, 2004) that found that people who use keyboard shortcuts are more likely to work with and/or around others who use keyboard shortcuts. Lee and Barnard (1993) suggested an explanation for the relationship between working with peers who are efficient and efficient usage. In their research, participants attended a training session on how to use a word processor and then came back the next day to complete the test and transfer tasks using the word processor. The test task required the participants to edit one part of the document while referencing another part of the document. This task could be done efficiently with the split screen function. The training session was equivalent for all participants, but during the testing phase, the users were in one of two groups, i.e., the “Functional revelation” group and the “Problem revelation” group. During the testing phase, if the participants did not use the split screen function, they were given a prompt to encourage the utilization of this function. For the functional revelation group, they received a note describing an available function (split screen) that could help with the task and how that function was related to the task. However there was no description of the “general” nature of the problem. For the problem revelation group, they received a note that reformulated the problem and explicitly described features of the problem and solution. This description was very general and did not contain any reference to the software being used or of features in the software that could be part of the solution. It was thought that if the users could formulate an appropriate concept of the problem that they would be more likely to retrieve the split screen function as a solution to that problem. Participants who had the problem
reformulated during the test session were more likely to use the split screen function
during the transfer session than those who were provided prompts on the appropriate
function to use and how to use it. Lee and Barnard’s research suggests that observing and
working with others may provide opportunities for the relevant features of a problem to
become apparent and knowledge of these features may facilitate the use of the most
appropriate (efficient) methods of addressing the problem.

In a similar vein, Nielsen, Mack, Bergendorff and Grischkowsky (1986) observed
during a field study of software usage that there is often a “computer guru” in workgroup
settings. Gurus are individuals who know more of the features of the software (and how
and when to use those features) than their colleagues do. These gurus are used as a source
of troubleshooting information as well as a reference regarding features and best
practices, thus their availability may increase the likelihood that someone will learn and
use a more efficient method.

Peer-to-peer learning may have benefits beyond learning to use software more
efficiently. A study investigating learning outcomes in the workplace suggests that peer-
to-peer learning may have social benefits as well (Evans & Kersh, 2004). Specifically,
that when people are engaged in sharing knowledge with their peers, they feel good about
the fact that they are helping others.

Although the benefits of this type of learning are strong and consistent, there are
certainly drawbacks. For instance, not all peers actually know the most efficient methods
of using software. Furthermore, it is conceivable that if someone watches a peer doing a
task inefficiently, this may reinforce the utilization of the more inefficient method by the
observer. Another challenge with learning from peers is that when this type of learning
occurs during the normal working day, it can take two people away from their work, thus increasing the cost of learning the more efficient method.

Although there are drawbacks to learning from peers, the findings strongly suggest that this may be an effective method for people to learn to use software efficiently. Although these findings seem to be consistent in the literature, they have primarily been an aside, thus leaving many unanswered questions regarding this process.

Summary

Many issues are left unresolved regarding the efficiency with which software is used. The ones that appear to be the most important are the identification of how efficiently software is currently being used and factors associated with the adoption of efficient methods.

One of the specific aims of the current study was to investigate how Carroll and Rosson’s findings generalize to software and workers in the 21st century. Furthermore, this study will investigate if the patterns of behavior and observations Bhavnani and his colleagues made in their ethnographic study would generalize to a more “typical” office setting in which users are working with programs such as those found in the Microsoft Office suite (Bhavnani et al., 1996). A general theme throughout much of the work presented on the efficient use of software is the user’s motivation to change their knowledge or behavior. Thus, this research will investigate what motivates people to learn (or not learn) methods that are more efficient.

The second aim of this research was to examine how people learn to become efficient users with specific focus on the relationship between collaborative learning and expertise level. Much of the previous work investigating the efficient use of software has
focused on the interface. This has produced information regarding changes that could be made in the human (e.g., training) or in the machine (e.g., interface redesign). However, given the evidence presented that efficient behavior may be closely tied to collaborative learning, there is reason to believe that a systematic investigation into the working environment would provide insight on the role that learning from peers may play in how people become efficient users of software.

To address these aims, an ethnographic investigation of an office setting was conducted. The main objectives of this research were to investigate the degree to which inefficient behaviors occur and how people learn efficient methods of using software. The ultimate goal is to establish a theoretical framework describing the conditions under which software users progress from novice to expert. It is hoped that this framework will more clearly explain the paradox of the active user as described by Carroll and Rosson (Carroll & Rosson, 1987). Furthermore, it can be expected to inform the design of both software and training programs that will increase users’ expertise levels with software.
ETHNOGRAPHIC METHODOLOGY

Ethnography is a method traditionally used by anthropologists and is defined as:

“The study of people in their natural settings; a descriptive account of social life and culture in a defined social system, based on qualitative methods (e.g., detailed observations, unstructured interviews, analysis of documents)” (Bowling, 1997).

In these qualitative observations, the researcher may find patterns of behaviors and insights into the phenomenon under investigation. Andersen provides the following description of the ideal results of ethnography:

“It is the patterns and the patterning that the ethnographer is looking for and not simply a realistic, behavioralized description or natural history... the ordinariness is somehow rendered extraordinary and, yet, recognizable. The deeper patterns being played out, in and through the detail come to the surface.”

(Anderson, 1994, p. 158)

Thus, the utility of this approach for the researcher lies in the freedom to explore and discover. The researcher is put in the position of making observations that will lead to new theories and hypotheses as opposed to systematically and methodically testing hypotheses to prove (or refute) a theory.

The value of ethnographic methodology for the disciplines of human computer interaction (HCI) and especially computer-supported cooperative work (CSCW) became evident in the 1980s (Blomberg, Burrell, & Guest, 2003). It has been used successfully in
the development of a variety of tools, software, and communication systems (Bentley et al., 1992). One of the more successful uses of this methodology was done to inform the design of an air traffic control system in the United Kingdom (Bentley et al., 1992). Bentley and his colleagues worked with an interdisciplinary team of ethnographers and software developers to determine the design requirements for a new flight database. Through observations and interviews, the ethnographers collected information about how the air traffic controllers and their assistants did their work. This information was integral to the iterative design process utilized by the design team. The process involved the ethnographers making detailed observations of the working environment then discussing their findings with the software engineers at “debriefing” meetings. The goal of these meetings was for the software engineer to gain an intuitive understanding of the needs of the air traffic controllers. These meetings would also offer an opportunity for the software engineers to request information from the ethnographer about a particular task or process that would affect the system design. Subsequent to this, the ethnographers could go back into the field to conduct focused and informed ethnographic investigations, providing the details about the working the environment that the software engineers needed.

Through this iterative process, the ethnographers ultimately discovered that ‘teamwork’ was integral to the job of the air traffic controllers (as opposed to being helpful but not necessary). In that environment, it was not functional to have a traditional division of labor system where each team member has a specific task to do. To accomplish their work, it was necessary for team members to do whatever task was necessary at any given moment, regardless of his or her particular title. Furthermore, they observed that manipulating and re-ordering the flight-strips (small strips of paper with the
pertinent information about a flight) was an important and informative process and should not be automated. Subsequently, the flight systems database and its GUI were designed to incorporate these findings. For example, one of the general guidelines for system design is to automate the more repetitive and mundane tasks. A task that would have normally fallen into this category, specifically manipulating and re-ordering the strips, was not converted to an automated task, as this would have seriously disrupted the air traffic controller’s workflow. Instead, the interface was designed to allow the controller the freedom to place the electronic version of the strip on the screen where he or she thought it should be.

Sellen and Harper (2002) provide another example of how an ethnographic investigation can provide unique insight regarding a working environment. In their book *The Myth of the Paperless Office*, they report their findings from several observational and ethnographic studies conducted with different businesses regarding the use of paper. To obtain information about how paper is used (e.g., when it is necessary, when it is helpful, and when it is a problem) they did observations, conducted interviews, and asked employees to keep diaries of the document and tools they used in their work. Through the information they gathered, Sellen and Harper were able to document how paper has different “affordances” than electronic devices. In other words, there are properties unique to paper that allows for particular and important behaviors and many digital devices do not allow for these same behaviors. For instance, because paper is light and mobile it is possible to position and re-position several documents on a desk in such a manner that the user can compare, sort, and/or stack the documents easily. They observed
this activity during their ethnographic investigation and it served to explain why people still prefer paper documents sometimes even when digital ones are available.

Ethnography has also provided important insight for researchers as evidenced in Bhavnani et al.'s (1996) ethnographic study of architects using CAD software. Through this study, they were able to explain the lack of increased productivity associated with the use of these programs. In this environment, the users were simply not aware of most of the functionality of the programs and this pattern of behavior would not have been apparent without observations of and interviews with the users.

Traditionally, ethnography involves field research conducted over an extended period, often for months or even years. Although these lengthy observation periods can allow the researcher to gain a uniquely deep and thorough understanding of a culture, they may not always be necessary for HCI research. David Millen (2000) suggested that with the use of multiple observers, key informants, and a limited research focus, it is possible to obtain rich and complex information about the work setting and "contexts of use" for any system design in much less time. Interviews are also an important source of information and are seen as an invaluable tool when conducting ethnographic studies. Appropriately done, interviews not only inform the research, but they can help identify where and when observations should occur (Blomberg et al., 2003). This use of observations in conjunction with interviews allows the researcher to probe deeper into a particular area of interest and elicit more details. It can also lead to the discovery of unexpected associations between behaviors, tasks, and people.
Current Study

This research involved an ethnographic study of software use in a workplace environment. The focus was on how efficiently people use software, differences in efficiency, and factors related to these differences. The study occurred over a four-week period and consisted of observations and in-depth interviews of employees at a large energy company in Houston, Texas. Subsequent to the interviews and observations, an on-line survey was administered to a sample of employees with the company. Employees were informed that participation in the observations, interviews, and/or survey was voluntary and that the anonymity of the employee would be maintained at all times. Before describing how the observations, interviews, and survey were conducted, a brief description of the working environment as well as a general profile of the employees and some of the issues they must deal with will be presented.

General notes about the setting

The employees interviewed and observed all worked for the same division within a large “in-house” information technology consulting (ITC) company that developed and supported applications for clients within the larger corporation. The areas in this division were: product development for collaboration tools; consulting services for math modeling projects, database management, and technology support; and application development. Furthermore, the business model of this company was such that almost all employees were on a billable hour system where the majority of their work time needed to be billed back to a certain project or client.
As a rule, the employees in this division were all very intelligent, highly trained, and motivated. The manager’s comment regarding the employees of this group was that, “When I found out who I would be working with when I took this job, I realized that they were all my heroes. These are great people who do great work.”

All of the employees performed the primary functions of their jobs on a computer and used several different programs on a daily basis, sometimes up to six or seven programs at a time. Programs were used to facilitate or support a majority of the communication and collaboration efforts of the employees and because of this, the employees often needed to work with a program while they were communicating with another employee. Furthermore, given that the parent corporation that ITC was within was an international company, there were employees in almost every continent. Therefore, much of the collaborative communication occurred across several different time zones and most employees in this group had to deal with issues associated with communicating and collaborating with other employees all over the world. This created challenges in several areas, specifically when scheduling teleconferences with others who were in different times zones (e.g., China). Another challenge arose with regard to running teleconferences. The leaders of these meetings had to: 1) maintain the focus of the meeting on the agenda of the meeting, 2) be sure to include in the discussion those who were not co-located with the rest of the group, and 3) operate the programs necessary to run the meeting. This was just an example of some of the issues associated with the some of the work done in this division. It is presented to illustrate the fact that these employees regularly experienced situations where technology (here, telecommunication) introduced problems in addition to solutions. These types of
problems were important to consider when defining efficiency within this working environment.

Coincidentally, within the last year this division had implemented a behavioral based safety (BBS) program designed to reduce the number of injuries that needed to be reported to the Occupational Safety and Health Administration (OSHA). For this group, the primary injuries reported to OSHA were repetitive stress injuries (RSIs). To reduce these types of injuries, employees needed to reduce their mousing activity and the number of keystrokes they made in a day in addition to maintaining good posture and mouse position with their hands, wrists, and arms. The BBS program involved: group stretch breaks; new equipment for all employees (e.g., keyboard, mouse, etc.); use of work monitoring software that reminded employees to take breaks from the keyboard and/or mouse; and peer consulting and monitoring regarding practicing “safe” keyboarding and mousing. There were corporate goals associated with this program and the manager of the group placed a high priority on safety in and out of the workplace. Due to this program, the employees in this division had a heightened sensitivity regarding being efficient with their software as efficient methods would reduce the number mouse clicks and keystrokes they needed to make in a day.

*Observations and Interviews*

Each employee had his or her own office on the same floor of a traditional office building. All observations and interviews took place in either the employee’s office or in a meeting room used for teleconferences.

There were eight people observed and the observations lasted approximately two to three hours. During the observations, the researcher watched the employee while he or
she worked and periodically asked questions regarding specific tasks or procedures. Approximately five of these hours occurred during meetings and these meetings always included telecommunications of some sort, whether the employee was in his or her office and called in to a phone bridge or the employee was in a conference room with other people and others phoned into the conference. The primary conference room had dry erase boards on three sides, a computer projector, Internet access, and a speaker/microphone telephone system for teleconferencing.

The interviews were open-ended interviews conducted with ten people and each interview lasted about one and a half to two hours. Interviews occurred in the employee’s office and consisted of the employees answering questions regarding their computer training, programs they used the most, how they learned these programs, and how they learned new things about the program.

**Survey**

To further investigate some of the patterns of behavior documented during the observations and interviews, a short on-line survey was administered to all of the employees of the division where the ethnographic information was collected. There were 170 employees in this division and 52 responded to the survey. The employees were located in both Houston, Texas and in San Ramon, California. To increase the response rate, the survey was intentionally kept short and had 20 questions thus it took approximately five minutes to complete. The questions asked which programs employees specialized in (Specialty) versus the program they spent the most time using (Time). Other questions asked the respondents to indicate how important it was to them to know the features of the programs and to use the different programs efficiently. A final group
of questions asked respondents to provide information about how often they learned new features and from whom.
RESULTS

In this section, descriptions of the observations are presented with enough information to allow one to understand the type of work each individual did but without enough information to allow for identification of the individual. In this same vein, third person pronouns may or may not reflect the employees' actual gender. The environments where meetings occurred are described as well as the observations made at two in-house training programs. This is followed by a discussion of the themes that emerged from these observations and the survey results.

For the sake of the observations presented here, inefficient use of software was considered the use of features and/or methods that took more time than another method of accomplishing the same task. In principle, a metric for efficiency could be the minimum amount of time necessary to complete a task divided by the total time someone took to accomplish the task. For instance, if the minimum amount of time necessary to change the width of three columns in a spreadsheet programs was 2.5 seconds and a user actually took 3.5 seconds to complete this task, he or she would have an efficiency index of .714. As the total time to complete the task came closer to the minimum time necessary to complete the task, the efficiency index would approach 1.0. Although it was not possible to calculate this metric during the observations, generally, this metric was considered when a behavior was considered efficient or inefficient.

Finally, screens shots of several of the programs used by the employees are provided in the Appendix. These images along with a description of the images are presented to provide context for the descriptions of how the employees were using these programs.
Field Notes

S1. This individual was a “team lead” in a group that maintained network infrastructure and developed collaboration tools. Before becoming a team lead, she was a programmer. Her job required her to do a lot of coordinating and planning. As a team lead, she tracked the number of billable hours for each of the individuals who reported directly to her and their availability for future projects. In a given day, S1 worked with other team leads and product managers and, when she worked on developing a collaborative tool, she would work with the users and the developers.

S1 participated in one hour of interviewing and three hours of observation over 2 different days. In general, she considered herself a very efficient user and reported that she uses keyboard shortcuts quite frequently. Interestingly though, S1 did not use the most frequently use keyboard shortcuts, specifically those for cut, copy, and paste. Based on the observations, it appeared that she was not quite aware of how much she used the mouse when she could have used the keyboard instead.

During the one-hour interview, S1 indicated that she preferred to learn a new program by first having a short initial formal training session consisting of a general orientation to the program. Specifically, in this training she would be interested in finding out what the program can do and how the interface was laid out. Subsequent to this, she liked to work on the new software with an actual project to learn how to move around in it. S1 reported that she used the “Tips and Tricks” files frequently and found this the best way to learn the features of the software that were not apparent on the interface. If she encountered any problems that she could not resolve herself, she would go to a co-worker or to the developer (if the project was developed in-house). S1 added that when she had
been involved in the development of a training program for new products, she would try to take the perspective of a new, naive user so the training protocol would be more helpful for a novice with the program.

In a given day, the programs S1 reported using the most were (in order): Outlook, Internet Explorer (primarily for Web-based applications), PowerPoint, Excel, Word, and Access. During the two-day observations, she primarily used Outlook, Excel, Word, and ERoom (a Web-based collaboration tool). There were four sources of information that prompted her to perform certain tasks: receiving e-mail, getting a telephone call, someone dropping in making a request, and reviewing the notepad where she kept meeting notes and items needing to be resolved. For instance, she was prompted to make a phone call to someone from an e-mail she received. Another example occurred when two men came to her office and asked about how to get administrative privileges on one of their computers. S1 showed them the Website where they could get temporary privileges and when one of them discussed needing permanent privileges, she said that temporary privileges were sufficient for what they needed. She knew why he needed the privileges and answered based on this knowledge. From the observations it was clear that much of the information (e.g., procedures, schedules, knowledge of project status, etc.) S1 needed to do her job was in her head. On a daily basis, she was involved in planning and negotiating based on the information in her head in addition to the information available to her through the corporate intranet. According to her, she used Excel to organize this information, Outlook to communicate, and the Web-based applications as managerial tools.

The first hour and a half of observation was in the morning and S1 was primarily going through her e-mails in Outlook. Once she read an e-mail, (or scanned it to know if
she needed to read it completely) she would mark the e-mail as read using a keyboard shortcut (Ctrl Q). S1 reported that she had tried using the feature in Outlook that marked the message as read after she viewed it for a few seconds. However, with that option on, the first message in the Inbox was always marked as read when she opened Outlook and thus she had missed messages in the past because she thought she had read them. Interestingly, even though she indicated that she marked all e-mails as read once she had determined if they needed action, she still had over 50 old (over a week old) unread messages in her Inbox. Furthermore, she had over 400 e-mails in her Inbox (see Appendix 1 for an example of an Outlook Inbox). When asked about this, S1 said that she stopped filing e-mails in her Inbox because she had trouble finding a filing system that worked. She had tried filing by project, month, etc. but nothing seemed to work, so she essentially kept them all in her Inbox and sorted them (by the sender, date, etc.) to find any she needed. Furthermore, since filing e-mails was not a billable activity, she did not want to dedicate an appreciable amount of time to it.

At one point, S1 was sending an e-mail to her direct reports and while she struggled to recall all of the people in her group, she mentioned to herself that she should make a distribution list for this group. At that point, her direct supervisor came in to observe whether S1 was using “safe” keyboarding and mouse positions (these observations were associated with the BBS program). S1 mentioned to her supervisor that she was about to build a distribution list for her direct reports. Her supervisor mentioned that she did not use distribution lists because she did not think she would remember the name she gave the group. S1 told her about the auto-fill feature in Outlook and how this offered suggestions if you typed in just the first few letters of the group (or e-mail) name.
Because her supervisor did not know about auto-complete, S1 showed her how to turn it on, at which point the supervisor reported that she would turn that on immediately and jokingly added that she would maybe even try a distribution list. Interestingly, although S1 knew about the two features of Outlook mentioned above, when she was scheduling a meeting, she went to a paper calendar to find out if a particular date was on a holiday instead of using the holidays function in Outlook. This function would populate her calendar with the national and religious holidays she chose, thus eliminating the need to refer to another document.

It also became clear after supervisor left that S1 had not made many distribution lists before. S1 started making the distribution list and struggled with the interface quite a bit. She was not sure how to move names from one window to the next (see Appendix 2 for an example). S1 moved her mouse very quickly around the window and clicked almost every button, closed and opened the window several times, and still struggled. Her interaction with the interface seemed to be almost random and she did not necessarily read the buttons she clicked. Once she did figure it out, she selected and moved each name one at a time. When asked if she was aware that she could select more than one at a time and move them all at once and she was not.

Although S1 kept her calendar current in Outlook, she also had a paper calendar that she took with her. She used it not only for marking when important events were, but also as a sort of filing system. For instance, she had a printed e-mail in her paper calendar. This e-mail had an agenda for an off-site meeting she was attending later that week.
When using Internet Explorer, S1 would open a new window with the keyboard shortcut Ctrl-N and used the mouse for everything else. At one point, S1 was using ERoom to work on an agenda (in Word) for an upcoming meeting. She opened the Word file through the ERoom interface and made several edits to it. When she was ready to save and close, she realized that she did not remember whether she had opened it in the edit or read mode. If she opened it in the read mode, she would not have been able to save her changes so S1 copied the changes she had made to the document, tried to save the document (found out that she had opened in read mode) and then closed the file. S1 opened the file again in edit mode, pasted her changes, and then saved the file again to the ERoom.

Most of the Word files S1 used during the observation were shared files, i.e., someone else in the group had made the original documents and she was editing them to communicate information to the associated group. In one document, S1 was editing a list of people’s names and the names were in several columns, delimited by tabs. S1 struggled with editing the format to get the names to line up correctly and it was not clear if she knew that the list of names could have been converted to a table for easier editing. In the same document, S1 tried to edit a table and could not figure out how to insert cells. She was clearly not aware of the tool bar available in Word that provided all of the features she needed and ultimately gave up on the attempt (see Appendix 3 for an example of the Word toolbar). She seemed to be almost impatient with the process and, similar to when she was trying to build the distribution list, was quickly moving around the window, clicking everything she thought might make a difference.
S2. This individual participated in a one-hour interview and three hours of observation. S2 was a database manager/consultant and was primarily responsible for servicing and maintaining clients' databases. She worked with the backend of these databases exclusively and the customers were responsible for developing/buying their own front end for accessing the data in these databases. She was involved with implementing and monitoring the BBS program in the division and coordinated reporting activities from the employees to upper management. She was also part of an intensive mentoring program designed to develop potential leaders from within the company.

Her job as a database manager primarily involved communicating, troubleshooting, and writing and issuing SQL commands. Her primary methods of communicating were through e-mail and the phone. Similar to most others in the organization, she used Outlook more than most other programs during any given day. S2 also used a text editor (Notepad) and Telnet (to connect to the databases). To manage the databases, she used two different DOS based programs as well as two programs with GUI interfaces, one she used for the first time during the observations.

As part of the BBS program, S2 had actively tried to reduce the number of keystrokes she made. Associated with this effort, she made several changes to her Windows settings. S2 started using the Microsoft Office Toolbar (see Appendix 4 for an example) to access the programs she used frequently instead of having to go to the start button every time. She reported learning this by playing around with the computer. S2 also changed her interface to a "one-click" environment so she only needed to click once on items on her desktop to open them. She learned this feature through an e-mail from a colleague.
S2 described another time that a colleague told her of an efficient method that she
did not know about. The colleague observed her using the computer and saw her lock her
workstation by clicking Ctrl + Alt +Delete and then move her mouse to the “Lock
Computer” button and click it. He told her that she did not have to use her mouse to lock
the workstation and that after issuing the key combination of Ctrl, Alt, and Delete, she
could press enter and this would lock the workstation. S2 had been using this method of
issuing this command for several weeks and surprisingly, at the time of the observations
she had not realized that by pressing enter she was essentially “clicking” on the
highlighted button (see Appendix 5 for an example). She assumed it was another key
combination she had to memorize.

S2 learned the key combinations for cut, copy, and paste by looking over
someone else’s shoulder. She reported using these somewhat frequently and was
observed using them when editing e-mail messages. Interestingly, she primarily used
copy, paste and delete, that is, she would copy a section of text, paste it to a new location
in the e-mail, and then return to the original text and delete it. Since S2 started using
keyboard shortcuts, she reported that she started noticing them in the dropdown menu,
but did not usually initiate learning them.

In addition to learning the shortcuts in the Windows programs by watching others,
she had also learned from others some shortcuts with the DOS-based database
management software she used. Particularly, she would copy and paste with the mouse in
the DOS environment. One of the more surprising observations with S2 was that when
she was typing text (an e-mail or commands) she would not look at the keyboard very
frequently at all, but as soon as she went to use a keyboard shortcut, she would look
down at the keyboard. Furthermore, when she shifted from working on a DOS-based program to a GUI interface, her entire posture would change somewhat. In the DOS-based programs, S2 was fluid and relaxed, although in the GUI environment, her movements seemed more forced and much less fluid. It is important to note that she used the keyboard almost exclusively (by necessity) in the software she “lived-in” (where she had the most of her primary billable hours) and almost exclusively used the mouse for the GUI-based programs, with the exception of the shortcuts mentioned previously.

In contrast to her efforts to reduce the number of keystrokes she used, S2 had made no effort to learn the features of her new company issued mouse in spite of the fact that she had received an e-mail with information about how to use her new mouse. She reported that she just did not want to bother with it.

During much of the observation, S2 was troubleshooting with other people and communicating with them either via telephone or via e-mail. As part of this process, S2 would often reference other e-mails or run reports using SQL commands. During one of the e-mail exchanges, she ran a series of SQL commands to obtain information regarding the status of a particular database. To do this, she opened a file in Notepad where she had previously saved the commands necessary to run this report. She copied and pasted the commands to the database management program and then ran the report. Ultimately, S2 ran the report three different ways and communicated the results of all three in an e-mail. She spent quite a bit of time formatting the e-mail so her findings would be clear (e.g., having the results in different colors for the different methods of obtaining them, etc.).

In the process of addressing this issue, S2 used a new program for the first time. She had already installed it on her computer, but had not used it yet. She was trying to
use the new program to run the same report that she normally ran with her own commands saved in a text file. The program used a GUI and she moved around in it very intentionally as opposed to the more haphazard exploration observed with S1. There were tabs on the left of the screen with different categories of operations. S2 looked through the words on each of these tabs and when she was not sure which was the right one, she went through all of the tabs to see what functions were associated with each category. S2 did this a few times until she realized that she needed to connect to the database through the program. She found where to do that, connected to the database and then went to the tab that had the report she was looking for. S2 ran the report and found that those results matched the results from her own script. She seemed comforted by that—as if the fact that the programs’ results matched hers gave her more confidence in the program. She reported that when she learned a new program, it was usually in this manner. She would install the program onto her machine and then when she needed it for a particular task, she would play around with it until she figured it out.

After she had run the report using the GUI program, she mentioned that although she was glad to know the new program, she would continue using the DOS based program for database management as well. Her reasoning for this was that she wanted to maintain her knowledge and skill with the SQL command language. She worried that if she exclusively used a GUI database management system that puts the commands behind the scenes she would forget the command language.

S3. This individual participated in a one and a half hour interview as well as three hours of observations. He was primarily responsible for the corporate Web casts (e.g., the chairperson’s annual international Web cast) done by the company. Some of these Web
casts were for communication and some were for training. His training was in Audio Visual and he had worked with video productions like commercials and training videos. He knew some programming and primarily used the Office suite for his job in addition to programs that supported graphics and media.

S3 seemed very cognizant of the different approach he took for learning different types of software. He discussed that his “specialty” was associated with the video applications, but that the vast number of his hours at work were spent in Office programs, particularly Outlook. Although he used Outlook the most, he reported that he only learned something new about Outlook when the need arose. S3 was not expected to be a subject matter expert in Outlook nor did he want to be. In fact, he reported that he actively tried to limit the amount of time he spent learning new features or methods in Outlook because he believed it would be more beneficial to spend that time honing his specialized skill.

Nevertheless, there were situations where he judged it worthwhile to invest time learning new features in Outlook. S3 described two times when he decided to learn new features in Outlook because he had seen those features use effectively by others. One instance was when he needed to poll several people regarding a particular issue. S3 had seen someone else send an e-mail using the “voting buttons” feature in Outlook and he thought that this would be an effective method of getting the information he needed. (The voting button feature in Outlook allows someone to send e-mail to a group of people asking them to “Vote” for their choice. The receiver gets an e-mail with buttons showing the available options and then can click on one of these buttons to indicate his or her choice. Outlook then sends this information back to the originator of the e-mail.) S3
explored the interface, learned how to do this, and had used it periodically since then.

Another example occurred when he needed to use the “Out of Office” feature in Outlook. When activated, this feature of Outlook will send a message in response to any incoming e-mails letting the sender know that the recipient is out of the office. Again, he had seen it used before and had a need for it so he explored the interface of Outlook until he figured out how to do it. There was another time that he needed to change where the information in his Outlook program was being stored, specifically, he needed the information to be stored on the company’s server instead of the PC in his office. He figured out how to do this by exploring the interface. Interestingly, he learned this, not because he had seen someone else use it, but because he had a need and thought that Outlook would be able to do it.

There were also features of Outlook that S3 had used previously, but currently did not need. For instance, Outlook had an address book feature that would allow the user to keep a list of people’s addresses, phone numbers, e-mails, etc. The user can create their own Address book that is stored on their PC and/or they can use a global address that is stored on the company’s server. The human resources department maintained the information provided in the global address book and was, for the most part, current and correct. At one time, he kept his personal Outlook address book up to date. However, he ultimately realized that most of his contacts were within the company, thus he could use the global address book and not spend time maintaining a separate address book. Another feature of Outlook he had tried was the Notes function. This was a feature of Outlook that allowed the user to make small electronic notes that could be referred to later and filed in electronic folders. He eventually found that writing things down worked better for him.
than having the notes available electronically—he mentioned that there seemed to be something about the process of writing things down that helped him process and remember the information better.

Although S3 was not necessarily interested in learning features of Outlook, during the observations it became clear that he had developed several distinct and seemingly efficient methods, of using the features he knew. For instance, at one point S3 received an e-mail asking him to respond to a question he had answered many times before. To create the response, he went to his Sent mail folder to locate a previous response he had made to a similar question. Thus, he used the Sent mail folder as a repository of responses he had made to different questions, similar to a list of Frequently Asked Questions (FAQs). When asked directly about this, he mentioned that at one time he had a more formal process for maintaining a public list of FAQs but that he had not put any effort into maintaining it lately. Although S3 had not thought of using the Sent mail folder as more efficient than a public list of FAQs, it is conceivable that using previous responses as a template for future, and personally addressed, questions could have been a more efficient way of providing personalized customer service to his clientele. This is particularly relevant given that many of his clients were senior level management who may have responded more positively to personalized responses.

During the observation, S3 drafted and sent several e-mails. While he was drafting the e-mails, he would spell-check what he had written, sometimes two or three times. This was in addition to having Outlook spell-check the e-mail before he sent it. He used the menu to issue the command and did not know that he could issue this command by pressing the F7 button. Once he learned about being able to use F7 to issue the spell-
check command, he said he would start using it and seemed enthusiastic about having a faster way to issue this command that he used so frequently. However, in the next two or three e-mails he sent, he did not use it and even realized he did not use it, but not until after he used the mouse (with the menu) to issue the command. A week later, when he was asked about whether he had adopted using F7 for spell-check, he smiled sheepishly and said that he had not. He said that he would often think about using it but only just as he had finished doing a spell-check, not before. He added that he would often do spell-check mindlessly as he was composing an e-mail and that this behavior was more something he did while he was thinking—his mind being on composing the e-mail, not on the spell-check.

In contrast to how he issued the spell-check command, there were several instances where he used the keyboard instead of the mouse to issue commands. For instance, when he was looking in the Sent mail for the response he had made previously to a particular person, he sorted e-mails by sender and then typed the first letter of the name to jump to it instead of moving down the screen with the scroll bar. He also used the used the keyboard commands of shift and arrow down to select text and Ctrl-C and Ctrl-V for copy and paste. It is important to note that these methods could be used in most of the Windows applications and were not specific for Outlook.

Although he would learn new features of Outlook only when he needed them, for programs associated with his specialty, he would actively and continuously make efforts to learn new features and discover best practices (or efficient methods) with those features. He reported that these programs were generally more intrinsically interesting to him and he tended to experiment more with them. For example, when first learning a new
version of Photoshop, one of the programs associated with his specialty, he would look at all of the items in the drop down menus to see what functions were available. If he were not familiar with a particular function, he would take time to find out what it was, even if he did not necessarily need to use it. He explicitly said that he would never do that sort of thing for a Microsoft Office product.

Another of the programs associated with his specialty was Microsoft Producer and he was expected to be a subject matter expert within the company for this tool. Because it was necessary to change some of the defaults on it to satisfy their customers’ needs, he had to get to know this tool inside and out. Furthermore, he trained people on how to use this program. There were primarily two types of people he did training for: 1) those who were not multimedia specialists but had a need to use this technology and 2) those who were one step removed from the information source and were interested in multimedia. For the first group, he would train them on what Producer did and how it did it. For the second group, the training focused primarily on best practices with Producer as opposed to the teaching the functions themselves. Another differentiation he made between the two groups of users was that some were novices and others were inexperienced. He described novices as those who did not have a basic understanding of the fundamental concepts in Producer. For instance, they usually did not understand the timeline structure of a video. In contrast, inexperienced people were those who understood conceptually how to build a video but did not know the specific features of the Producer software.

S4. This individual was a math modeler and she participated in a one-hour interview as well as a three-hour observation session. The program S4 primarily used was a modeling
program that was almost exclusively run through a GUI and did not require much typing at all. One of the unique aspects of this employee was that she first used a PC computer when she started her current position. Previously S4 had worked exclusively with Macintosh and Unix-based computers and with those computers, she used the keyboard to issue commands. However, in her position at the energy company, she used the mouse almost exclusively for all of the programs she needed. Furthermore, in her previous work environment, S4 worked around others doing similar work and they would share “Tips and Tricks” on how to use the software programs and, in contrast, for her current job she worked alone in an office and was the only one in the local office who used the math modeling program.

When S4 first started using the PC she was so unfamiliar with it, she had to call a friend to ask how to check her e-mail. S4 was embarrassed to tell anyone that she did not know how to use the PC since she was hired as a computer expert. She learned how to use the PC by trial and error and never went back to using the keyboard for any commands. S4 did mention that she had gone to training for a software program and during that training they pointed out some of the shortcuts during the training. However, she had not adopted any of these.

During the entire observation, S4 was using modeling software for her current project. This project was somewhat different from other projects her group had done because they had competed against outside bidders for it. The group, and thus she, was motivated to show that an in-house product could be as good as that provided by an outside consultant. She had previously used this particular software for another project,
but was having to learn new functions for this project. There were very tight deadlines associated with this project and she had to learn a lot about the program very quickly.

S4 had not had any training on the program and learned the software by exploring the interface and calling a 1-800 technical support number. She reported that the technical support was “pretty good” and that she called so frequently, she and some of the customer service agents were on first-name basis. Although there was another person in the company who used this software, that person was in California and primarily available via telephone. Traditionally, when they started a project, S4 and her colleague discussed what each of them knew and could do on the project and the software, so she had a good idea about what her colleague could and could not help her with.

At one point S4 was trying to determine how to output data from a particular object in the model to an array that she could manipulate. She worked over an hour to learn how to get the information she needed. She essentially had to determine some of fundamental logic of the software system for herself. During the hour she used the help file, the glossary, the support Web site, and the manual to get the information she need. She would talk to herself while she was doing this and seemed to be testing different hypotheses about the logic of the interface and terms. For instance, at one point S4 said, “If I click here, then it should show me the values I need [clicks button] [pause...] but it doesn’t so that must mean that this is not the right screen…” Although this process was long and painful, it was obvious that she was learning a lot about the program during the process. Through the trial and error process, she discovered functions she did not know previously. An example of this was observed when she tried several different types of graphs to get the data she needed since she had not be able to get the data in a table
format. S4 was finally able to get the data exported to a graph. Many of the graphs she tried were new to her, thus if she had not spent this time exploring the interface, it may have been much longer before she learned about these features. Although she learned a lot about the software through the trial and error process, she was not necessarily learning how to use the features she was learning efficiently. An example of her not using efficient methods was illustrated when she was deleting four items (out of five) from a window. To do this she had to select the items and then delete them. Instead of using the more efficient method of selecting the four items she wanted to delete and deleting them all at once, she deleted each one individually (similar to Appendix 2).

S4 mentioned that for this project she was going to have to learn how to format reports in Excel. Given that her group had competed against outside bidders for the project, S4 needed to deliver the reports in a format that was more polished than some of the previous reports she had done. She had not worried about formatting Excel spreadsheets with other projects because there were usually very tight time constraints and other projects did not have a high priority placed on the presentation of the reports. For this project, the customer had shown her examples of the types of document they wanted, thus she was planning to spend more time on it for this project. Although S4 was familiar with macros and formulas in Excel, she was not at all familiar with how to create formatted spreadsheets. As opposed to how she learned the modeling program, she said that most of the employees in the office knew more about formatting spreadsheets, so she planned to ask her peers if she had trouble doing something.

S5. This employee worked with the training group and had diverse responsibilities. During three hours of observation, S5 described her responsibilities and used several
different programs to accomplish these tasks. S5’s job was to identify and develop best practices for the Corporate IT employees and to develop and deliver training programs to teach those best practices. For example, she developed a training program on the best practices for creating a business proposal. In addition to this, she managed many of the daily/administrative issues for the group (e.g., reconciling billing codes, ordering supplies, etc.). She primarily used the Office suite for her job, and Outlook was the program she used the most.

S5 reported that she knew of many of the features in Outlook but that she did not use most of them. There were many features she had tried (e.g., Journal - the feature in Outlook that tracks the documents a user has open during a given day) but thought they were not helpful because she did not necessarily know how to use them correctly. Thus, she did not continue using them. Although S5 did not use many of the features in Outlook, she had developed efficient strategies that helped her organize information in Outlook. For instance, at one point S5 received an e-mail regarding a particular project. She changed the subject line in the e-mail to mark it as associated with that project so she could file and retrieve it more easily.

S5 explained that in Outlook, there were features she knew but did not use because she did not know when to use them. Conversely, in Excel and Word, there were features that she knew and did not use for other reasons. For example, in her previous job, S5 became very familiar with Excel (she described herself as a power user), however in her current job it was usually not necessary for her to use much of the functionality she knew in this program. Therefore, in this example, she knew the features and when to use them, but did not have the opportunity to use them. Another example of a feature that she
knew, knew when to use, but did not use was revealed when she shared an interesting story regarding the auto formatting features of Word. Auto formatting is a feature in Word that involves the program automatically reformatting a page based on a template. Often this will include Word adding page breaks particularly with large tables. S5 reported that the auto formatting made it very difficult to use templates they had previously created in Word using tables. S5 considered herself “fairly robust” in the Office tools and between her knowledge and the Help Desk they could not “fix” the templates and had to rebuild the entire document. Because of this, she avoided creating templates in Word and was leery of using the auto formatting features in documents she would need to share with others. In addition to the Office programs, she used a Web-based SAP application and spent more time with this program than the others (with the exception of Outlook).

At one point during the observation, her co-worker came in and asked her to help him figure out how to access his computer remotely (i.e., when he was at home). S5 used a Web application through the company’s Website to try to do this. While she was exploring the different Web pages, she very rapidly, and seemingly randomly, clicked on many of the clickable features of the page. The two of them spent about 30 - 40 minutes trying to figure this out and at one point, they were in two different offices trying to simulate a remote access to the computer. Ultimately, they were not able to resolve the issue and had to call someone they knew would have the answer. This person was able to provide the answer they were looking for and provided other information they needed, although they did not know they needed this information. This interaction was interesting for a few reasons. The first was that it was a good example of intentional peer-to-peer
learning that resulted in the person learning how to do what he needed to do, although not ultimately from the person he sought out. Another interesting observation in this process was that both of them learned from each other information about the corporate remote dial-in interface. In fact, between the two of them, they were able to figure out the information provided on the Website and most of the information they needed to submit for remote access. Indeed, the information they ultimately received about how to access computers remotely was not available on the corporate intranet so it was not that they missed the information they needed, the information was not there.

Based on the observations and conversations with her, in contrast to the previously mentioned employees, there did not appear to be a program that was strongly associated with her specialty (i.e., developing and delivering training). She did use several efficient methods that could be applied to most Windows programs: Ctrl-C and Ctrl-V for copy and paste; shift + arrow to select files in Explorer; and Alt + tab to move between open programs. When asked how she learned these methods, S5 reported that she did not remember how she learned the first two, but that she learned the Alt + tab method when she saw someone doing it in a meeting. Interestingly, to close a window, S5 would double click the top left icon of a window as opposed to using the more efficient method of single clicking the top right of the window.

S6. This employee was also in the training group and had as his primary responsibilities developing and conducting training programs. In addition to training responsibilities, he facilitated projects that the training group was conducting. All of the 3 hours of observations of this employee were in the “war room” during teleconferences (the
conference room described previously). During the two meetings observed, there were three or four people in the room and two to four on speakerphone.

For both meetings, S6 had the meeting agenda in a Word document, projected onto the screen for those in the room and available via Meeting Place for those who were attending the meeting remotely. Meeting Place is a program that allowed someone to have their computer desktop securely visible via the Web for those who were attending the meeting remotely. He specifically referred to the agenda during the meeting and often used the mouse like a pointer to draw attention to a particular part of the document. In addition to the Word document, in both meetings, he accessed Websites for the group to discuss. Again, in this situation, his use of the mouse was very effective for pointing out particular elements of the Web pages for both those participants in the conference room and those attending remotely.

S6 claimed to be “computer challenged” and reported that he relied on his peers to help him when a problem arose or when he did not know how to do something on or with a particular program. Nevertheless, he used several efficient strategies, two of these strategies being similar to those his colleague used. Specifically, he used Ctrl-C and Ctrl-V for copy and paste and for all e-mails and appointments he indicated which project the item was associated with for easier filing and retrieval as well as color-coding his Outlook calendar. For Web pages he needed to access during the meetings, he would put the shortcut to the Web page on his desktop and in his favorites. Given his proclaimed status as being “computer challenged,” it was a bit surprising to see how effective and fluid he was while running the two meetings. He was able to easily move between two and three different programs while running the meeting and documenting the discussion.
that occurred during the meeting. However, after the meeting, when he was observed using an Instant Messaging program, his discomfort with computers became more obvious. He struggled with adding a new person to the list of people he could instant message even though he had been using the program for "a while now." He did not seem to have any idea at all of how it worked, and seemed to be randomly clicking every clickable item on the screen in an attempt to find the person in the company list and add her to his buddy list. He was not successful in this endeavor and ultimately e-mailed her instead.

An interesting moment occurred during the meeting—as S6 was editing the meeting agenda, the group started to discuss where the document would be saved so they could all have access to it. Several suggestions were made, but none were sufficient for the needs of the group. No one in the group seemed to be aware that the ERoom program, developed in-house, could meet the needs they articulated during the meeting. Evidently, the application had been presented to the group but for a different use so it did not occur to them to use it for this function. Although it is difficult to determine exactly why this happened, it is a bit surprising that it was difficult for this group of information technology professionals to see possible applications of a software beyond the application presented to them.

S7. This employee was a manager and his main responsibilities were to manage projects and the resources (i.e., people) necessary to complete those projects. During two and a half hours of interviews that occurred over two days, he described his previous training, his current job, and how he used programs. Before being a manager, he was involved in project implementation as well as the development of training for some of the programs
he helped to implement. The implementation process required application development and when he was involved in this process, he would often initiate learning new features about the programs he was supporting. However, he reported that with his current job he no longer initiated learning new features of a program very frequently.

When discussing the programs he had used, he grouped them into three categories: Microsoft Office programs, internet-based tools for managers, and programs he had been involved with developing and implementing. The former two types of programs, Office and the internet-based tools were the ones he reported using the most frequently, although he reported not spending much time at all on the latter type of software.

He described Excel as the software that was integral to his job. He used it to track what people were working on and their availability in conjunction with a Web application that provided information about customer requests. He learned how to use Excel approximately three years ago when he needed to manage large amounts of information regarding people and activities. He used the program primarily for monitoring and managing (sorting, filtering, etc.) these data and did not often use many of the other features. He reported that he did not often see other people use Excel and felt that his knowledge of the features in Excel was sufficient to do what he needs to do.

Although Excel was the program he saw as fundamental to his job, like many of the other employees in this group, Outlook was the program he used the most. He learned it by exploring the interface, from asking others, and by watching others. For example, he learned that he could have a window for both his Inbox and Calendar open at the same time after he saw someone else with their Outlook arranged in a similar way. He
estimated that he learned something in this manner about three or four times a year. One of the efficient methods he used with Outlook that he did not learn from someone else was using the read/unread status of e-mail as a way of identifying whether the information in the e-mail needed to be followed up on later. Specifically, e-mails that had information he would need to reference later were left as unread and filed in a mail folder in Outlook. However, those e-mails that were not as important remained in his Inbox and the read/unread status of those e-mails did not necessarily mean anything regarding the status or importance of the e-mail. These were just e-mails that he did not want to take the time to process (delete) thus, he had left them there.

The second group of programs he described was the Web-based manager’s desktop tools and he used these infrequently for salary issues, employee information, etc. He had done some computer-based training for these but he did not use these often so he found it hard to remember how to use them. He would often try “hunt and peck” to remember how to do the functions and if he had trouble using them, he would call a peer and they would usually hunt and peck their way through the program together. He mentioned that he did not think it was worth his time to become efficient with these programs because he did not use them very often. Specifically, he said, “(It was) Low on the value chain to become an expert with these applications.” It was interesting that he had thought about the fact that he is not efficient with the Web-based managers tools and had justified this lack of efficiency. Conversely, it seemed that there was room for efficiency improvements in his use of Excel, but he did not mention giving any thought to this.
The last type of software he described were those where he had been involved with the development and/or implementation of the software. In these situations, he had a more in-depth understanding of the programs and how they worked. He often would have very in-depth training associated with these programs and this gave him a broader picture of the applications. When he used these programs in his current job, he found himself making evaluations/observations of how they had been implemented or presented. This observation seemed particularly relevant because it illustrated that S7 previously had a job where he needed (and wanted) to know many if not most of the features and how to best use those features.

_S8 and S9._ These two individuals were developers and their jobs were essentially to create new programs for the group or the group’s clients. During a one-hour interview, both of them shared information about the programs they used and the work they did. Many of the programs they developed were Web-based and the tool they primarily used for programming was the .Net development environment. Both of them had started using this programming environment within the last twelve months and had different experiences concerning the adoption of it.

S8 spoke of the improvements in the .Net program and how she thought those improvements helped her productivity. In contrast to this perspective, S9 talked about how he saw the benefit of using the new tool but that had experienced “growing pains” adjusting to the way the new tools worked compared to the tool he was using previously. Both of these programmers were using the same tool before they started using .Net, but they were using it in very different ways. Thus, S9’s transition to the new tool had been more difficult than S8’s. S8 mentioned that for her, the new tool was easier to use.
because it had brought together all of the features she used in different applications, thus it was a very welcome change for her. However, S9 had to learn a lot of new processes and concepts in order to use the basic tools in the .Net environment. S8 also mentioned that during the training for the .Net tool, people seemed to respond differently based on their background, specifically the type of programming they did and what software they had used previously.

An important and fundamental difference between S8 and S9 was that for S8’s projects, she used the .Net tool almost exclusively, however, due to the nature of his projects, S9 switched between several programs in any given week. He described this as very frustrating for him because he might spend one entire day in the .Net environment and then the next day not touch it at all, but instead be in a different development environment. S8 further reported that as she used .Net, she learned more of its features and thus was able to give up using the other programs she used previously for those features (e.g., with database access, .Net would let her access databases without using SQL servers or Access). These two developers’ experience with adopting a new software is very interesting because it illustrates that if someone’s knowledge structure of a task is similar to the knowledge structure used by the software, that person may have an easier time learning the software than if there is a mismatch between the two.

Regarding Office programs they used frequently, they both spoke of their frustration with the communication process that occurs with Outlook. They clearly experienced e-mail as a distraction from their job and did not like getting instructions this way, particularly if it required them to decipher what the e-mail means. For instance, they found it frustrating when they received e-mails with an exchange on it and were asked to
“figure out what the customer wants.” They wanted to just be “left alone” to do their job. They had much less to say about the software itself and seem to be inclined to use it as little as possible, primarily because they did not want to communicate with people via e-mail, not necessarily because they did not like the program.

Given the length of the interview, they were not able to share much more information about other programs they used, but S8 did report that she used Excel to “organize her thoughts” when she was building an application. For example, she would use a spreadsheet to help her track the fields in an application that needed to be visible and searchable as well as which fields should appear after a search.

They also both shared some information about their use of the keyboard to issue commands. S9 reported that he did not use many keyboard shortcuts—maybe four or five on regular basis. He did not actively search for shortcuts unless there was a specific task that he would be repeating several times. If this were the case then he would explore the interface to learn the shortcut and use it to reduce the amount of time required to issue the command. He started using the keyboard to issue commands when the Send button was accidentally disabled in his Outlook program requiring him to use the keyboard shortcut for sending e-mails. S8 no longer used the keyboard to issue commands because of a time when she was using the keyboard to issue commands in Outlook and accidentally sent a message before she was ready.

Training Programs. Over the last twelve months, the company had developed a series of training programs designed to increase employees’ awareness of features in the Office software that could increase their productivity. Observations were made at two of these trainings—Advanced Outlook and 60 tips in 90 minutes. These training sessions provided
information about the relevant and useful features of these software programs and the attendees seemed receptive to the information they were getting. The trainer would tell the attendees of a particular feature and then would show how to issue that command, using the drop down menus almost exclusively to issue the commands she was teaching.

In the Advanced Outlook training session, at least three administrative assistants attended and they asked very specific questions about tasks they needed help with. The questions they asked and the way they framed the questions seemed to indicate that they thought about the tasks they do in Outlook differently than the trainer did. For instance, when the trainer was describing a feature in the Outlook calendar associated with scheduling meetings, one of the attendees asked if the feature worked in a similar fashion when she was creating an appointment for someone else. Specifically, if she (the administrative assistant) were requesting a meeting in her supervisor’s name, would the feature work the same as it did when she was requesting a meeting for herself? After the training, the attendees voiced to each other the concern that they may not remember the things they had just learned when they got back to their desk. The attendees also asked if it would be possible to have training sessions occur in their office at their computer. They seemed to think that they would better be able to learn some of the features presented in the training if they could directly apply them to tasks they normally do. Immediately after the training (and some during the training), they shared with each other information and possible opportunities to use the features they were learning.

Although both of the training programs provided important information and tools, these features were not presented in a manner that articulated how their use could increase productivity. The training seemed to be more of an attempt to increase the
number of features the trainees knew about, not necessarily an effort to teach them how
to utilize these features to increase their efficiency.

Efficiency with software

Three main themes emerged from the observations and interviews. 1) Employees
approached learning and using software differently depending on whether it was
associated with their specialty or interest. Examples of software associated with a
specialty are: computer programmers might use the Microsoft’s .Net development
package to support their expertise in programming; or Webcast developers might use
Microsoft Producer to support their specialization. This was observed with many of the
interviewees and some overtly stated that they did not see a benefit in being efficient with
some software although other software they wanted to know as much about as possible.
2) Employees reported learning software primarily by exploring the interface and through
their peers. All of the people interviewed said they initially learn most software programs
by exploring the interface to learn how to do what they needed to do. Furthermore,
anything they knew beyond what they learned through the interface, they learned through
peers and colleagues. 3) When learning a new program associated with their specialty,
employees seemed to be using their understanding of the task to understand how to use
the software. However, when learning a program not associated with their specialty, they
seemed to be more concerned with figuring out how to accomplish the task and they did
not try to understand the logic of the software.
Levels of Efficiency

A primary aim of this study was to examine the degree to which people were currently using software efficiently in the workplace. The short answer to this question is “it depends.” Specifically it depends on the type of software the person is using. As mentioned previously, respondents did not have a specific and consistent attitude toward learning software programs and becoming efficient with all of them. This is in contrast to what would be expected based on the goal orientation literature. If this theory applied to learning software, employees would have displayed one of two different orientations toward becoming efficient with all software they use, i.e., learning orientation or performance orientation. This theory assumes these orientations are consistent across learning domain and type of task (Dweck, 1986). As detailed below, the present data contradict this prediction from goal orientation theory. In fact, users adopted different orientations for different programs. Moreover, although there is certainly evidence of the production bias seen in these results, the bias was not universal (Carroll & Rosson, 1987). If a program were associated with their specialty, it was much more likely that employees would stop the task they were doing to learn a new feature or method, but for other types of software, they would continue using a method they knew to be inefficient.

Observations. A good example of how the employees approached learning differently depending on the type of software was provided by S3 who said that after he installed a new version of the program he specializes in, he would look through each of the drop down menus to see the new commands and features available in that software. However, for Outlook, he intentionally spent the minimal time necessary to learn only those
features he thought he needed. This is in spite of the fact that he spent the vast majority of his time at work using Outlook and used the other programs only once or twice a week.

Another example of this was observed with S2 who had spent several hours setting up her desktop so she could easily and quickly access the programs she needed to do the tasks associated with her specialization (i.e., not having to go to the “Start” menu to open the programs). However, she had not spent any time at all getting to know the features available on the new mouse the company had provided. This is particularly interesting because her motivation behind making the programs easier to access was primarily due to the BBS program focusing on reducing RSIs. The new mouse was provided as part of this program and was advertised as having features that would help to reduce RSIs. Thus, she seemed to think it was worth the time to reduce the number of clicks required to open the programs she used for her job, but was not interested in how to use the mouse to improve her efficiency. Furthermore, she had not put much energy at all into ways to reduce her keystrokes with the program she spent most of her time on, specifically Outlook.

A third example of this was cited by S7 who said that with his current job, he rarely initiated learning new features of the programs he worked with, but that when his job was application development, he would do this frequently. When he was doing application development, he felt that it was his job to know as much about the program as possible and he enjoyed the process of learning what the programs could do. Interestingly, this person also said that Excel was the program integral to his current job, but that he was aware that he did not use all of the features in it that could apply to him. He was a manager and used Excel to organize and manage projects and the time and
resources (e.g., people) allocated to each project. At first glance, this example would seem to contradict the finding that people are more likely to learn a program more fully if that program is associated with their specialty. Upon closer examination, however this may not be the case. Excel is a very powerful and flexible spreadsheet program, but was not necessarily designed as a project management tool. Thus it may be that he did not perceive knowing all of the features of Excel as fundamental to doing his job as a manager. However, when he was an application developer, it was important to have a detailed level of knowledge about a program he was “rolling out” to the company.

Although, generally people seemed more motivated to increasing their expertise with software associated with their specialty or interest, there were those employees who utilized efficient methods for all types of programs. For instance, S2 kept a copy of the SQL commands she used most frequently in appropriately named text files. When she needed these commands, she copied and pasted them from the text file to the DOS window instead of rewriting them every time she needed them.

Another best practice was observed with S3, who used his sent mail folders as a repository of FAQs. When he would receive e-mail requesting information he had provided to someone else, he would copy the response from a previous e-mail, edit it to address the unique needs of the current client, and send it on. Although this may not seem as efficient as maintaining a list of FAQs available for people to read without contacting him, S3 was often working with very senior executives and their assistants, therefore the need for personal customer service was high.

A third example of someone utilizing best practices was observed with S7 who used the read/unread status of his e-mail as part of his filing system. When he received e-
mails needing attention later (e.g., billing issues or quarterly reports), he would put the unread e-mail in the appropriate folder in his Outlook program. When the time came that he needed to access that information, he could go to that folder and know that only the unread ones needed to be reported for that particular period. After the report was submitted, he would mark the e-mails as read and start over again for a new reporting period.

Although S7 used an efficient method of organizing important e-mails, he, similar to most of those who were observed, had more than 400 e-mails in his Inbox. Some of these were marked as read and some were not. Although this may not seem like a big problem, because e-mails in the Inbox are saved on the corporate server, having this much e-mail in each person's Inbox unnecessarily takes up a huge amount of server space. The storage space on servers is a huge expense for all the divisions of this company, thus the practice of periodically emptying this out is a large cost savings for the company. Interestingly, several employees made the comment that cleaning out their Inbox was not a billable activity, so they did not put a high priority on spending time doing this.

Survey. Respondents were asked to provide the software that they spent the most time using in a given day and the software they associated with their specialty (or that they considered fundamental to their being able to do their job). They were given the following examples to clarify what was meant by a software being fundamental to their job: "... if you are a computer programmer, the software you use to support your expertise might be a program like .Net, but you may spend most of your time using MS Outlook.
Another example would be an accountant whose expertise is supported by a program like Peachtree, but may spend most of his or her time using MS Excel.”

Based on the responses to the surveys, the respondents were categorized into three groups: those who had a clear specialty program that was different than the program they spent the most time on (Clear Specialty), those who spent the most time using their specialty program (Same), and those who did not have a clear specialty program (No Clear Specialty). The No Clear Specialty group consisted of people who had indicated that their specialty software was the Office suite. Given that this can include several different software programs, it was not clear which of the programs they considered integral to their job so these individuals were categorized separately from the other two groups. Table 1 provides a list of the programs the respondents listed as their specialty program (Specialty) as well as the ones they spend the most time using (Time). Over half of the respondents (N=30) reported spending more time on a program that was not their specialty program, 17 reported spending most of their time working with their specialty software, and five of the respondents had no clear specialty program.

Table 1 reveals that in all categories and groups, Outlook is used more than any other program. Moreover, regardless of the specialty program, most respondents spent most of their time using Outlook. Nevertheless, the employees use a wide range of programs varying from development platforms for programmers (e.g., Visual Basic .Net) to applications in the Office suite (e.g., Excel). However, it is important to note that it cannot be determined from this survey how much more time the employees spent on one program over another.
Table 1.
*Frequencies of programs used by respondents as a function of the group and the type of software.*

<table>
<thead>
<tr>
<th>Specialty</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear specialty (N = 30)</td>
<td></td>
</tr>
<tr>
<td>Browser-based applications</td>
<td>1</td>
</tr>
<tr>
<td>Crystal Enterprise</td>
<td>1</td>
</tr>
<tr>
<td>DBArtisan</td>
<td></td>
</tr>
<tr>
<td>Exchange</td>
<td>1</td>
</tr>
<tr>
<td>Exceeds/hummingbird</td>
<td>2</td>
</tr>
<tr>
<td>Excel</td>
<td>4</td>
</tr>
<tr>
<td>IBM Content Management On Demand</td>
<td>1</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>1</td>
</tr>
<tr>
<td>Lotus Notes/Domino</td>
<td>1</td>
</tr>
<tr>
<td>Office</td>
<td>5</td>
</tr>
<tr>
<td>Microsoft SQL Enterprise Manager</td>
<td>1</td>
</tr>
<tr>
<td>Oracle</td>
<td>4</td>
</tr>
<tr>
<td>Outlook</td>
<td>7</td>
</tr>
<tr>
<td>PowerPoint</td>
<td>1</td>
</tr>
<tr>
<td>SAS</td>
<td>1</td>
</tr>
<tr>
<td>TextPad</td>
<td>1</td>
</tr>
<tr>
<td>Tibco Active Enterprise</td>
<td>1</td>
</tr>
<tr>
<td>Visual Basic .NET</td>
<td>1</td>
</tr>
<tr>
<td>Visual Studio .NET</td>
<td>3</td>
</tr>
<tr>
<td>Word</td>
<td>1</td>
</tr>
<tr>
<td>Same (N = 17)</td>
<td></td>
</tr>
<tr>
<td>Excel</td>
<td>1</td>
</tr>
<tr>
<td>Lotus Notes/Domino</td>
<td>4</td>
</tr>
<tr>
<td>Oracle</td>
<td>1</td>
</tr>
<tr>
<td>Outlook</td>
<td>8</td>
</tr>
<tr>
<td>Visual Foxpro</td>
<td>1</td>
</tr>
<tr>
<td>Visual Studio .NET</td>
<td>1</td>
</tr>
<tr>
<td>No Clear Specialty (N = 5)</td>
<td></td>
</tr>
<tr>
<td>MS Office</td>
<td>6</td>
</tr>
<tr>
<td>Outlook</td>
<td>4</td>
</tr>
</tbody>
</table>

*Specialty indicates program employees specialized in and Time indicates program employees spend the most time using in a given day.*

Based on the observations that people approached becoming efficient with software differently, the responses were examined for those who had clear “specialty”
software that they did not spend most of their time using. Since there was no specific expectation regarding the other two groups, no analysis were done for these groups—however the results are presented for descriptive purposes.

Figures 2 and 3 provide evidence that employees who have a program that is associated with their specialty approach that program differently than they do other programs, even the program they use most frequently. Figure 2 shows the distribution of responses to the question regarding how important participants thought it was to be efficient with the two types of software. Participants in the Clear Specialty group rated the importance of being efficient higher for the program associated with their specialty than for the program they spent this most time on, $t (29) = 2.01, p = .054$. In contrast to this, for both of the other groups, there were virtually no differences between the importance of efficiency for their specialty program versus the program they used more often.
Figure 2. Box plots for the responses to the question on importance of efficiency as a function of Software type (Time and Specialty) and Group (No Clear Specialty, Clear Specialty, and Same). In addition to the median, each box plot shows the mean ± 1 standard error. Responses based on a five-point scale with 1 being “Not Important at all” and 5 being “Extremely important.”

Figure 3 presents the distribution of responses to the question regarding how important participants thought it was to be knowledgeable about the features of the two types of software. This figure shows that individuals in the Clear Specialty group thought it more important to know most of the features of the program associated with their specialty versus the program they used the most, $t (29) = 4.56, p < .001$. Consistent with the responses to the questions regarding efficiency, for both of the other groups, there were essentially no differences between the importance of knowing the features of the specialty program versus the program they used more often.
Figure 3. Box plots for the responses to the question on importance of efficiency as a function of Software type (Time and Specialty) and Group (No Clear Specialty, Clear Specialty, and Same). In addition to the median, each box plot shows the mean ± 1 standard error. Responses based on a five-point scale with 1 being “Not Important at all” and 5 being “Extremely important.”

Learning

Another aim of the current study was to identify how people are currently learning to be efficient with a specific focus on the relationship between collaborative learning and expertise level. Without exception, the employees reported learning most software programs on their own. Normally, they reported that they would be assigned a project that required using a new program and, while working on that project, they would explore the interface to learn how to use it. This was the case regardless of the complexity of the software and occurred for software as sophisticated as a modeling or
database management program to one as straightforward as a Web application for accessing employee information.

*Observations.* Although in some instances formal training was available, often employees had not had any training when they started using new software. S1 said specifically that she preferred using software for a while to become familiar with the basics of it before she had any formal training. She thought if formal training occurred after someone was already familiar with the program, the training could focus on teaching how to maximize the program's features instead of presenting information that is easy enough to learn without training.

By spending time exploring the interface, employees appeared to get to a level of knowledge with the software that was sufficient for them to perform the tasks associated with their jobs and they learned any additional knowledge through interactions with peers. Moreover, the employees reported learning both features of and best practices with software from peers. This peer-to-peer learning was sometimes intentional, (i.e., one person would ask another person for help) and sometimes accidental or incidental (i.e., one person would see someone perform a task in way that he or she was not familiar with and then would ask about it). Given that the interfaces are not necessarily designed to teach efficient methods, learning through peers may be the only opportunity some people have to learn best practices procedures and concepts.

People also seemed more comfortable with requesting help from a peer or a supervisor than asking a subordinate or referring to a manual. When asked what they would do if they had a problem with a program that they could not resolve themselves, most reported that they would ask a peer. Some mentioned that they might ask their
supervisor, but none mentioned learning from a subordinate and only one person mentioned that she would use the manual. Furthermore, they usually had an idea of whom they would go to for specific types of help. For instance, there was one person in the office that was considered the subject matter expert on Outlook and another who was mentioned by several as being an expert on the more complex and obscure features in Excel.

An example of incidental learning occurred during an observations with S1. When her supervisor came in, S1 mentioned that she was about to build a distribution list in Outlook. At that point, her supervisor had an opportunity to explain why she does not use this feature of Outlook (i.e., because she would not be able to remember the name she gave the distribution group). With this information, S1 was able to tell her supervisor about the “Auto-complete” feature in Outlook. Entirely by accident, S1’s supervisor learned a feature in Outlook that she would not have known otherwise.

S7 described another example of incidental learning when he mentioned that he had been trying to figure out how to have his Inbox and Calendar open at the same time in Outlook. Coincidentally, he saw someone who had his/her Outlook view setup in a similar fashion to the one he wanted. This prompted him to ask that person how to get that type of view. The two of them spent a few minutes discussing what he needed and then the other person explained that he could use the “Open in New Window” function to accomplish what he wanted.

A third example of this type of incidental learning was described by S3 who reported that at one point he needed to poll several people about a particular issue. He has seen someone else use the “Voting buttons” feature in Outlook, so he knew it was an
available feature. Though he ultimately figured out how to do it by himself, he was made aware of the feature through a peer.

The number of times employees reported learning a feature or method through a peer (either accidentally or intentionally) are too numerous to list, but many if not most of them were associated with Outlook or general Windows features or methods (e.g., using Ctrl-X, Ctrl-C, and Ctrl-V for cut, copy, and paste). This maybe because people had the most to learn in these particular programs, but was more likely due to the fact that people had more opportunity to see someone using these programs since they are the most frequently used.

Survey. Table 2 shows how many features respondents reported learning in the last six months, both for the Specialty and for the Time programs. Most people reported learning 1 – 3 new features in the six-month period of time and this did not differ for the type of software or the group. Participants reported learning these new features from their peers or online for both types of software and in all three groups (Table 3). Most people were aware of someone in their group who knew a lot about the programs they were using and reported that they would feel “Comfortable” to “Very Comfortable” asking this person for help with the program. These results support the observation that people learned new features of the programs they worked on through their peers and provides additional information that these individuals are learning from the Web as well (possibly through a search engine like Google or with the Online help tool).
Table 2.  
*Frequencies for: “How many times in the last 6 months have you learned something new about _____?”*

<table>
<thead>
<tr>
<th></th>
<th>Clear Specialty</th>
<th>Same</th>
<th>No Clear Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specialty</td>
<td>Time</td>
<td>Specialty</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 – 3</td>
<td>19</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>4 – 6</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7 – 9</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>&gt; 9</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

*Responses for Same/Time were the identical to those for Same/Specialty and thus are not reported.*

Table 3.  
*Frequencies for: “The last time you learned something new about _____, how did you learn it?”*

<table>
<thead>
<tr>
<th></th>
<th>Clear Specialty</th>
<th>Same</th>
<th>No Clear Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specialty</td>
<td>Time</td>
<td>Specialty</td>
</tr>
<tr>
<td>Peer</td>
<td>15</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>Supervisor</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Subordinate</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manual</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Online</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

*Responses for Same/Time were the identical to those for Same/Specialty and thus are not reported.*

Observations on learning software

Although the focus of this research was on the efficiency of software use, during the observations, a pattern of behavior became apparent that was unexpected and worth
describing. This pattern of behavior was an interesting intersection between the two observations already reported (people approached learning software differently and they learned software by exploring the interface) and became apparent when some employees were observed learning new software or new features of software they already were using.

Observations. The first two examples, from S2 and S4, were of women learning programs that were directly associated with their specialty or interest. In general, they were very careful and thoughtful with what buttons they clicked and why. They would talk aloud to themselves and although they clicked almost every button on every screen, this clicking was not random, but more experimental. Through S4’s dialogue, it was apparent that she was attempting to determine the logic of the software. This process was tedious, but also very educational, although not about best practices or efficiency. It is conceivable that when she learned how to do something through exploring the interface in this manner, this became the way she did that task and she would not necessarily go back later to find more efficient ways of accomplishing the same thing. At one point S2 was using a program for the first time to perform a task she had performed many other times using another program. She explored the interface of the new software slowly and intentionally, reading labels aloud and adding comments to herself regarding what terms were familiar to her. She was overtly mapping the wording of the functions in the new software to those of the previous software. These employees appeared to be intentionally learning a mental representation of the structure of the program and its functionality. They were concerned and focused on how different commands related to each other and
made note of features new to them even if they were not necessarily associated with the particular task they were trying to accomplish.

In contrast to this behavior, in two other situations where S1 and S5 were observed learning something new, their behavior was primarily that of randomly “clicking buttons to see what happens and maybe one will do what I want.” These individuals did not read what was on the buttons they were clicking, nor did they appear to be thinking about what action would occur by clicking the buttons. The programs they were using were not ones associated with their specialties. They were Outlook and a Web-based application that allows employees to access their computers remotely. In these two situations, the goal appeared to be to find the sequence of steps necessary to complete the task as quickly as possible. There did not appear to be any effort put into learning associations between terms or the structure of the software.

It is important to note that these observations, although interesting, certainly do not warrant generalizations regarding what type of information people want to know about different programs. However, there were other comments made that support the idea that the employees’ mental representation of programs may play a role in how they learn and use those programs.

For instance, in a discussion with S8 and S9, they described their experiences with adopting a new tool (program) for building programs. S9 commented that he knew that the new tool was beneficial but that adjusting to it was difficult because many of the procedures in it were new to him. S8 mentioned that for her, the new tool was welcome and easier to use because it brought together all of the features she had been using in different applications. Although both of these individuals had previously been using the
same program for development, because of their different backgrounds, they had been using the program in very different ways. S8 had a knowledge structure of the process that more closely matched that of the new program and thus had an easier time adopting the new program. S9 had not only to become comfortable with the new interface, he also had to restructure his knowledge of the task.

These observations suggest that people may attempt to develop a set of knowledge structures and/or a mental model of the software if it is associated with their specialty and may only want procedural knowledge for other programs. Although much of the research on using knowledge structures in training has been equivocal regarding its benefits, these observations indicate that people might benefit from some explicit information regarding both the structure and processes of the program (Stout, Salas, & Kraiger, 1997; Sumner, Bonnardel, & Kallak, 1997). Moreover, this perspective could explain some behavior that may seem inefficient on the surface, but in fact, serves the purpose of facilitating the development of a mental model and knowledge structure of the software. For instance, calling tech support may take less time when trying to learn how to perform a particular task, but the learning that occurs while exploring the interface would not occur if tech support were used instead of exploring the interface. A particularly interesting example of this can be made with regard to using a GUI interface for issuing SQL commands. When the GUI interface is used, it relieves the user of having to know the SQL commands and thus may reduce the time it takes to issue these commands and the errors that could occur. Thus, someone who uses the GUI regularly may eventually forget the SQL command language, and thus their knowledge of that process. If troubleshooting needs to occur or if the GUI program is not operational, the
user who has forgotten the SQL language will not be as effective as someone who has continued to use the SQL commands as well as the GUI interface. This is another example of how a behavior that seems inefficient on the surface (i.e., continuing to use a less efficient method of issuing commands) may serve a larger goal.

Survey. Some of the observations just described suggest that when the employees learn a program supporting a task they were already familiar with, they looked to map the features and processes of the software to their own knowledge of the structure and processes of the task. If this is the case, then we might expect people to want different types of information for programs associated with their specialty than other types of programs. Tables 4 and 5 present the participants’ responses when asked about the information they typically want and receive from the Help Desk. As seen in these tables, the type of information they wanted and received from the Help Desk did not differ as a function of group or type of software.

Table 4.

<table>
<thead>
<tr>
<th></th>
<th>Clear Specialty</th>
<th>Same</th>
<th>No Clear Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specialty</td>
<td>Time</td>
<td>Specialty</td>
</tr>
<tr>
<td>Procedures</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Explanation</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>20</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Responses for Same/Time were the identical to those for Same/Specialty and thus are not reported.
Table 5.
Number of times respondents ranked 1st the type of information they received from the Help Desk.

<table>
<thead>
<tr>
<th></th>
<th>Clear Specialty</th>
<th>Same</th>
<th>No Clear Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specialty</td>
<td>Time</td>
<td>Specialty</td>
</tr>
<tr>
<td>Procedures</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Explanation</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>16</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Responses for Same/Time were the identical to those for Same/Specialty and thus are not reported.

Given these results, it is possible that the survey respondents do not approach learning software differently based on whether the program is associated with their specialty. However, these results may be more a function of the information the respondents want when they have a serious problem that they cannot resolve themselves. Because it cost 50 USD each time they call the Help Desk, employees may consider this a resource of “last resort.” Thus, the type of information they want from the Help Desk may be different from the kind of information they want when learning software. It is also possible that employees have developed an expectation of what type of information is available through the Help Desk (i.e., troubleshooting) and thus, this is the type of information they want.
Efficiency

Many of the findings from the current research were similar to those of Bhavnani’s ethnographic study of CAD users (Bhavnani et al., 1996), in spite of the differences between the nature of the work done in an IT consulting company and an architectural firm. Specifically, it was observed that people, for the most part, did not use simple strategies that can reduce the time and effort necessary for completing tasks. An example observed several times during the current study was the employees’ failure to use an “aggregate” strategy when performing the same task on multiple objects (Bhavnani & John, 1997). For instance, when S4 was building a graph in the modeling program, she needed to select four out of five field names in one text box (the field names were selected by checking the “check-box” beside each one) and move them to another text box. Instead of first selecting the four fields she wanted to move and then moving them all at once, she selected and moved them one at a time. An aspects of these users that was very different from the group that Bhavnani observed was their general level of sophistication with the use of software. Some of the observations Bhavnani and his colleagues reported illustrated a fundamental lack of awareness regarding using computer programs. A good example of this was the architect who would close and re-open the entire program when he wanted to close one file and open another (Bhavnani et al., 1996). Although the difference between these two groups may be due to the fact that one is a group of architects and one is a group of information technology professionals, it seems more likely that people are generally more knowledgeable about using computer software now than in the early 1990s when Bhavnani’s observations were made.
The similarity between this study and some of Bhavnani’s observations support the argument that people still do not use software efficiently. Indeed, employees generally spent less time learning how to become efficient with the program they spent the most time using—Outlook. Given the collective number of hours these employees used Outlook, the opportunity for timesavings was greater with this program than any other. However, the efficient use of software is not an all or nothing event. People have degrees of efficiency and the results from the current study clearly indicate that their level of efficiency differs by software. The finding that employees approached learning and using efficient methods differently for software associated with their specialty was not anticipated. Although anecdotal observations and the goal orientation literature (Dweck, 1986) suggest that an individual’s efficient use of software would be consistent across different kinds of software, the evidence presented here does not support this argument. The employees showed evidence of the production bias Carroll and Rosson describe (Carroll & Rosson, 1987) but it would appear that this bias is not universal. For software associated with their specialty, users are more likely to stop “throughput,” or working on the task, in order to learn something new about that software program.

One reason users may consider efficiency more important with their specialty software is that they are simply more inherently interested in the types of tasks done in this software. Although it is reasonable to expect that interest level had something to do with this pattern of behavior, it most likely does not explain all or even most of it. Another possibility is that people differ from each other in their absolute levels of efficiency but that within each person’s level of efficiency, they are more focused on being efficient with some software than others. For example, it is conceivable that one of
the reasons there did not seem to be a large difference in efficiency levels between individuals is the nature of the people in this setting. It could be that all of these employees have a learning orientation and since it is not possible for them to learn everything, they have to be selective with what they are going to spend their time learning. This is a very real possibility as all of these individuals were high achieving individuals just by the nature of their being employed with this group. However, if this is the case, their choice regarding which software they should focus on becoming efficient with could be considered less than optimal. Specifically, many of the employees observed and survey respondents reported spending most of their time using Outlook. However, they also reported that it was not as important to learn and use efficient features with this program. Thus, from an efficiency standpoint, the employees should place a higher priority on becoming efficient with the program they use the most. From a job performance standpoint though, this may not be the case. None of these employees were hired because of their skills with Outlook and none of them will have their efficiency with Outlook as part of their performance review. Given this perspective, the decision to not focus effort on becoming efficient with Outlook would seem a more reasonable (albeit misguided) choice.

This raises the question, “How do users decided whether they will stop work to learn (or even become better at) a more efficient strategy?” One employee indicated that if there were obviously a time benefit for learning an efficient method of using the software, he would take the time to do so. As an example, he said that he would make a point to learn the keyboard shortcut for a command if he knew he was going to issue the command several times in a row. Indeed Card, Moran and Newell (1983) found that users
would employ a dominant method of performing a task and would only deviate from this method if were “obviously inefficient” (Card et al., 1983, p. 151) to continue using the dominant method. This suggests that a user’s decision regarding whether to learn or use efficient methods may involve the consideration of elements over and above the time and effort required. Users may consider the importance of being efficient with a particular program in addition to the timesavings. Specifically, users seem to engage in a tradeoff calculation—trading off the efficiency gains with changing their behavior and the importance of those efficiency gains for each program. If this is the case, the metric for efficiency previously presented (minimum time to complete task/actual time to complete task) may need to include the importance of the timesavings as well if it is used to predict users’ behavior.

In this study, people placed more importance on being efficient and knowledgeable with software associated with their specialty versus other software. In other words, their motivation for efficiency was higher with their specialty programs than others. When this is considered in conjunction with Peres et al.’s (in press) findings that people’s behavior could be explained by the importance they place on that behavior, it may follow that people are more efficient with software that is associated with their specialty. It is important to note that these relationships are tenuous given the current findings and controlled experimental studies are planned to determine if this relationship holds. However, if a relationship does exist between the importance someone places on efficiency and the actual use of efficient methods, this would suggest that not only do people need to learn how to be efficient, but they also need to believe that being efficient is important.
Learning

Considering the amount of money and effort businesses have put into the development of different training programs and paradigms, it is more than a little ironic that all of the people observed for this study learned most of the software they use without any formal training. Though it is ironic that they do not use formal training programs, it is not surprising. Carroll and Rosson documented in 1987 that people want to get to their task as soon as possible. They would rather learn how to use the software to do their task while actually performing a task (Carroll & Rosson, 1987). Furthermore, many of the current programs are specifically designed so this type of “walk up and use” behavior is possible. These programs have extensive help files, available both within the system and via the Internet. It is also not surprising that the employees reported learning of efficient features and methods from their peers. This type of learning, whether described as collaborative learning or peer learning, is often referred to in the software efficiency literature (Bhavnani et al., 1996; Brown & Duguid, 1991; Carroll, 1997; Carroll & Rosson, 1987; Simon & Werner, 1996), but as mentioned previously, not specifically studied. Carroll and Rosson (1987) even acknowledged that people often went to their peers for information on how to use the text editors but they minimized the importance of this behavior when gaining expertise. Specifically they said that they “have no reason to believe that users will take the time and effort to find and consult human experts any more than they would a reference manual” (Carroll & Rosson, 1987, p.85). Findings from the current study certainly contradict their assumption. Their statement regarding peer learning also does not take into account the fact that in many work environments, employees are constantly consulting each other on all types of matters,
both work and non-work related matters. Because of this, much of the learning that takes place between peers occurs incidentally. Although it is difficult to predict or plan for these types of events, the learning that takes place during these exchanges is important and may have a larger impact on someone’s becoming an efficient user than is currently known.

Brown and Duguid offered an interesting perspective regarding the transmission of information between co-workers (Brown & Duguid, 1991). In their description of the how copy repairpersons exchange knowledge, the process of “story-telling” is presented as the mechanisms through which the problem machine is described. Using stories to explain how to perform a task may facilitate someone’s understanding of the problem because it puts the problem within a context that the learner knows. When users try to learn new features by themselves, they often have no context for the feature and thus may not be able to incorporate it into their current work. An example of this is the employee who reported that she had tried using the feature in Outlook that tracks the documents she has used during a given day. She had opened the window and tried to use it but was not sure what the “best practices” with this feature were, thus she never adopted it. She had taken the initiative to learn something new about the program she spends most of her time on, but because she had no context for the feature, it was not perceived as being helpful. Another relevant point brought up by Brown and Duguid (1991) is the issue of constantly changing circumstances. People’s jobs and the software they use are changing very rapidly. Utilizing others’ knowledge could be a sound and (possibly) efficient method of keeping up with these changes.
It may also be that learning from peers has many of the elements of the behavior modeling training described by Simon and Werner in their investigation of three different training methodologies (1996). A behavior modeling training environment allows the user to watch others do the task, do the task themselves, and get immediate feedback on how they performed the task. Collaborative or peer-to-peer learning situations can involve all of these elements and this may contribute to the appeal of learning in this manner. Linton and his colleagues attempted to generate a knowledge-sharing tool that simulates the type of sharing that often occurs between co-workers (Linton et al., 1999; Linton et al., 2000). The recommender system Organization Wide Learning (OWL) is an attempt to synthesize the collective knowledge among all of the employees in an organization and then to dissemination that knowledge in a targeted fashion to those who need it. A challenge for this system could be that it is missing critical parts of the collaborative learning process, i.e., the opportunity to see the new knowledge in action and get feedback when initially performing the task.

It would appear that learning efficient features and methods by observing peers serves several purposes. It exposes the observer to a new feature or method. It gives that person the opportunity to see the new method in action and in that process may provide information that the new method is more efficient that other methods. Observing someone else use an efficient method also provides the observer/learner with a context for the new feature. Finally, when someone observes another person utilizing a feature that the observer did not expect, the surprise of seeing the task done differently (particularly if it was done more quickly) may create a novelty effect that could increase the likelihood of the observer remembering and subsequently using the new features.
The training paradigm proposed by Bhavnani and his colleagues (Bhavnani, John, & Flemming, 1999; Bhavnani et al., 2001) could be a near approximation of the learning that occurs through peers, particularly since the class consists of students getting feedback on their performance as the semester progresses. Bhavnani and his colleagues designed the class to teach efficient strategies that would apply to several different software programs. Furthermore, they incorporated the four “components” of strategic knowledge into the class: knowledge of the strategy; knowledge of when to use the strategy; knowledge of how to execute the strategy; and knowledge that the strategy is general (i.e., can be applied across tasks and software). Thus, with the exception of the novelty effect, the format of this course appears to meet all of the purposes that are met through collaborative learning, i.e., learn the feature, see that the feature is efficient, and learn when to use the feature. Bhavnani appears to have developed a training methodology that is on the right track and the results from evaluations of this course bear this out. In one study, the instructor explicitly taught efficient methods to students in one group and only taught the features themselves to the students in another group. At the end of each session and at the end of the year, the students in the first group were more likely to use the efficient methods than those students in second group. Two challenge for researchers and software designers are (1) to determine how the type of information and guidance Bhavnani recommends can be infused into an exploratory environment that current workers are experiencing and (2) leverage peer-to-peer learning to facilitate the adoption of efficient use of software.
Other observations

The observations suggesting that people may approach learning software differently if they have previous knowledge of the task have interesting implications with regard to Carroll and Rosson's description of the assimilation bias (Carroll & Rosson, 1987). Carroll and Rosson presented the assimilation bias as a failure of metaphors. Specifically, that if someone came to a program with an existing metaphor for the task (e.g., a word processor as a "super" typewriter), learning was facilitated as long as the metaphor held. However, if there were a mismatch between the user's metaphor and the functioning of the program, learning would be impaired. For the users in the current study, there did seem to be an affect of having previous knowledge of the task on how they learned the software, but this did not seem to be associated with a matched or mismatched metaphor. Instead, users with existing knowledge of the task seemed to use this knowledge to guide their exploration of the interface of the software. Conversely, those users without knowledge of the task appeared to explore interface of the software somewhat haphazardly, almost randomly. Although the information available in this study is certainly not sufficient to make a general statement, the finding are nevertheless intriguing and warrant further investigation in a more controlled, experimental environment.
CONCLUSIONS

One of the unique aspects of using an ethnographic methodology is the opportunity to discover unexpected relationships and patterns of behavior. Indeed the two major findings from this investigation—that users approach learning software differently based on the type of software and that much of the learning that occurs is through peers—could not have been identified without going into the environment where people were using and learning these programs.

The insight from these findings indicates that people are not maximizing the power of the programs they have available to them. Furthermore, from these insights, it would appear that Carroll and Rosson (1987) made some profound assessments of how people go about increasing their efficiency with (or rather not improving their efficiency with) software. Their description of the production and assimilation biases are applicable to today’s working environment and thus seem to describe important characteristics of how humans interface with machines. Employees were more willing to stop their work and learn something new for software associated with their area of expertise than with other types of software. This suggests users may be evaluating more than the time associated with using particular methods when they are deciding whether they will adopt a new method. They seem to be weighing something that is unique to the content or the process of specific software and not something that can be generalized across software. In order to incorporate this type of information about the user into a user centered design process, it is necessary to understand more thoroughly exactly what users are evaluating in their decision regarding when to learn and/or use different features of software.
The findings from this study and the review of the literature also provide information that can inform and guide the training processes. Users continue to need information on what the features of the software are, but seemingly more important, they need to know how, when, and why to use these features in combination to efficiently conduct their work. Bhavnani and his colleagues (Bhavnani et al., 1999; Thomas & Foster, 2001) have made important efforts to this end and may have provided a model of how to teach efficient methods. However, this model is one of classroom learning and what is clearly presented from the users in this study and in the literature is that learning occurs primarily through exploring the interface and through peers. The challenge now is to develop a learning mechanism or method that will capitalize on the learning processes currently being employed in the workplace and use this mechanism to deliver the material developed by Bhavnani and his colleagues. Although this is a tall order, the findings presented here suggest that this may be an effective method of improving the efficient use of software.

An important point to consider is that the findings regarding people’s motivations to be efficient could conceivably apply to work patterns in general and not specifically to software use. However, because software is generally designed so people can explore and learn independently, it may be easier for someone to act on a motivation to acquire an efficient method with software than with other types of job related tasks.

One of the mantras for many who do work and research in HCI and in particular, usability is: “There is no such thing as user error. If an error occurs, it is most likely due to poor design.” Well, of course, there is such thing as user error, but the point was, and is, well made. Many would consider users’ not progressing to an expert level of
performance an error in the learning process. Designers can avoid these errors in the learning process by incorporating the needs and capacities of the user into the design of their programs. The work presented here suggests that there are more aspects of the user that need to be considered when designing software that incorporates the needs and capacities of the user—the motivations and mechanisms of how humans learn the type of complex procedural information contained in today’s software. For instance, in line with Fu and Gray’s findings (2004), designers could have the more efficient methods for issuing commands also be interactive and global, thus increasing the likelihood that people would select the efficient methods over the inefficient methods.

Although some may be ready to “blame the user” for the lack of transition to an efficient level of performance, until these aspects of how humans learn to interact with computers are incorporated into design, it is still appropriate to place this responsibility at the feet (or in the hands) of the usability specialist. However, it would help if the user placed higher priority on being efficient.
REFERENCES


Appendix A.

This is an Inbox in Microsoft Outlook Windows XP. On the left hand side of the screen is a list of mail folders where the user can move or file e-mails to for later reference. The pane on the right side is divided into two parts. The top half shows a list of e-mails in the selected directory or folder and the bottom half gives a preview of the content of the e-mail selected in the top half of the screen.

---

Chris,

Jim Slater states that we can expect 5 M Mcf/d each on the two wells that we are drilling at #A-323. There is no problem with separator capacity and we have a 20 MMcf/d sep. that they can use for their well. The reboiler that Fred has purchased for our new production is good for 15 MMcf/d. Generally that can be stretched about 25% and still have quality gas. If their well is more than 10 MMcf/d and our two wells come in at 10, this could be a problem but not a show stopper.

The compressor on the platform is good for 5 MMcf/d @ 600 PSI, so their well will have to be able to flow to HP or replace the compressor for a larger one. The two condensate pumps are rated for 200 bbls/d each. We may have to replace them if their wells are high yield condensate but we could tell them now. Water handling should be no problem.

In general, yes we have extra capacity.

Thanks, Jerry
Appendix B.

This is an example of the window in Outlook used to create a distribution list. The user must select names he or she wants to include from the list of names on the left hand side of the window and then move those names to the right window. A user selects the names in the left window by clicking on each name and then moves them by clicking on the "Members->" button in the middle of the window. Either the user can select one name at a time to move that name to the right hand box or it is also possible to select more than one name at a time (by having the Ctrl key pressed while clicking on the names) and move all of the names at once to the right hand box.
Appendix C.

The Word toolbar supports the creation and manipulation of cells, rows, and columns in tables. When the user clicks the icon on the bottom left corner, it lists several available options in Word tables (e.g., insert row, insert column, or insert cell).
Appendix D.

The Office toolbar has icons for different programs on it and the user can add the programs he or she uses most frequently or delete the programs not used frequently. It is always present on the user's desktop.
Appendix E.

This dialogue box appears when a user presses Ctrl + Alt + Delete to lock his or her workstation. Note that the "Lock Computer" button has a dotted line around the edge of it. This indicates that it is activated and if the user presses the Enter key on the keyboard, it performs the same function as if the button were clicked with the mouse.

<table>
<thead>
<tr>
<th>Login Information</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Logon Date:</td>
<td>4/16/2005 9:15:27 AM</td>
<td></td>
</tr>
</tbody>
</table>

Use the Task Manager to close an application that is not responding.