Beyond Command Knowledge: Identifying and Teaching Strategic Knowledge for Using Complex Computer Applications

Suresh K. Bhavnani
School of Information
University of Michigan
Ann Arbor, MI 48109-1092
Tel: +1-734-615-8281
bhavnani@umich.edu

Frederick Reif
Center for Innovation in Learning
Carnegie Mellon University
Pittsburgh, PA 15213
Tel: +1-412-268-5713
freif@andrew.cmu.edu

Bonnie E. John
HCI Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Tel: +1-412-268-7182
Bonnie.John@cs.cmu.edu

ABSTRACT

Despite experience, many users do not make efficient use of complex computer applications. We argue that this is caused by a lack of strategic knowledge that is difficult to acquire just by knowing how to use commands. To address this problem, we present efficient and general strategies for using computer applications, and identify the components of strategic knowledge required to use them. We propose a framework for teaching strategic knowledge, and show how we implemented it in a course for freshman students. In a controlled study, we compared our approach to the traditional approach of just teaching commands. The results show that efficient and general strategies can in fact be taught to students of diverse backgrounds in a limited time without harming command knowledge. The experiment also pinpointed those strategies that can be automatically learned just from learning commands, and those that require more practice than we provided. These results are important to universities and companies that wish to foster more efficient use of complex computer applications.

Keywords

Strategies, Training, Instruction, GOMS.

INTRODUCTION

Several real-world and experimental studies on the use of complex computer applications such as UNIX [9], word processors [12], spreadsheets [11, 8], and CAD [1, 4, 5], have shown that, despite experience, many users with basic command knowledge do not progress to an efficient use of applications. For example, Nilsen et al. [11], observed experienced spreadsheet users perform a task requiring a change of width of several adjacent columns with the exception of one. They found that most of the users modified the column widths one by one in order to avoid modifying the exception.

A more efficient method to perform this task is to aggregate all the columns (including the exception), modify their widths, and then modify the exception back to its original width. In applications where objects can be dropped from an aggregate, another method is to aggregate all the columns, drop the exception (e.g. through SHIFT-SELECT), and then modify the remaining set in one step.

Efficient strategies, such as the above used to handle exceptions, have two characteristics: (1) they are not easily acquired because they are neither suggested by the design of individual commands (such as SELECT, MODIFY), nor by the task [3]; (2) they are generally applicable in a wide range of applications. For example, the efficient strategies for modifying many elements with an exception are relevant in tasks ranging from moving files in an operating system, to modifying paragraphs in a web-authoring application.

We have come to believe that such *strategic knowledge* holds the key to efficient use [3, 5]. Because this strategic knowledge is difficult to acquire spontaneously just from a knowledge of commands and tasks, we hypothesized that users can benefit from explicit training which combines general strategic knowledge with specific command knowledge. This paper addresses two questions: (1) Can strategies be explicitly taught in combination with commands? (2) Can strategies and commands be taught in the same time as it takes to teach just commands and without hurting the learning of commands?

We begin by briefly describing general and efficient strategies, and show how these strategies were combined with specific command knowledge in the design of a training course for freshman students. In a controlled study, we compared the above approach to a traditional approach that focused on teaching command knowledge. The results show that strategic knowledge can indeed be taught to a diverse population of students without harming command knowledge, or taking excessive time. Furthermore, there was evidence for the transfer of knowledge across applications. The experiment pinpoints which strategies need special attention, and raises questions to be addressed in future research.

THE NATURE OF EFFICIENT STRATEGIES FOR COMPLEX COMPUTER APPLICATIONS

Complex computer applications, such as word processors and spreadsheets typically provide numerous general-purpose tools that can be used to perform a wide range of tasks. For example, aggregation tools allow users to group many kinds of objects such as words, paragraphs, pages, cells, files, and graphic objects in order to operate on them as a group. Similarly, formulas in a spreadsheet can be used to link cells to create many different and complex relationships.

However, the flexibility offered by such general-purpose tools comes at a cost. There are often many ways to combine them to complete the same task. This puts on the user the burden of knowing the alternate ways and to pick an appropriate one. For example, the spreadsheet task of modifying a group of columns with an exception, can be done in several ways, as described earlier. Furthermore, the knowledge of these alternate methods, and how to pick among them, is over and above the knowledge of using the individual commands. To perform the task efficiently, a user must know available methods for performing this kind of task (in this case, dealing with an exception in a group modification), and must then use the aggregation and drop tools in the correct sequence to complete the task quickly.

We refer to such *goal-directed* and *nonobligatory* methods as strategies [3, 13]. When we refer to *strategic knowledge,* we refer to the knowledge of the alternate methods to perform a task, and how to choose among them [5]. Our research results, which agree with several other studies [1, 8, 10, 11], have led us to believe that such strategic knowledge is difficult to acquire through command experience alone and, in fact, holds the key to the efficient use of complex computer systems. The first step, therefore, was to identify these strategies and understand the nature of the efficiencies they provide.

Identification of Efficient and General Strategies

The strategy for handling the exception before operating on a group is more efficient than changing the width of each column individually, because it exploits the iterative power of the computer. By specifying to the computer the precise aggregate of objects to modify, the user effectively delegates the complete iteration to the computer. This avoids the time-consuming and error-prone steps of changing the width of each column.

Because the power of iteration is pervasive in tools offered in computer applications, we identified several other iteration strategies [3]. All these strategies exploit different ways to set up aggregates of objects in order to create, reuse, and modify them efficiently. For example, an important strategy, when making many copies of an aggregate of objects, is to check that the original is correct and complete. Otherwise, the errors or incompletions replicate through all the copies and require subsequent corrections. These corrections can be time-consuming (if they are noticed at all) and can lead to yet more errors. GOMS [7] analyses of such iteration strategies estimated a reduction of between 40%-70% in execution time when compared to performing the same tasks without these strategies [1, 3, 4, 6].

Iteration
1. Reuse and modify groups of objects
2. Check original before making copies
3. Handle exceptions before/after modification of groups
Propagation
4. Make dependencies known to the computer
5. Exploit dependencies to generate variations
Organization
6. Make organizations known to the computer
7. Generate new representations from existing ones
Visualization
8. View relevant information, do not view irrelevant information
9. View parts of spread-out information to fit simultaneously on the screen

Figure 1. General and efficient strategies to exploit four powers of computers.

Of course, the power of iteration is only one of many that are offered by computers. Figure 1 shows a set of powers, and corresponding strategies that we have analyzed in detail. Besides strategies of iteration, they include strategies of propagation, organization, and visualization.

Propagation strategies exploit the power of computers to modify objects that are connected through explicit dependencies. These strategies allow users to propagate changes to large numbers of interconnected objects. For example, Strategy 4 makes the dependencies between objects "known" to the computer so that (1) new objects inherit properties or receive information from another object and (2) modifications can propagate through the dependencies. This strategy is useful in word processors through the use of styles. Here a user can create paragraphs that need to share a common format or to be dependent on a common definition; when the definition is modified, all the dependent paragraphs are automatically changed. Similarly, formulas in a spreadsheet can be linked to dependent data, or graphic elements in a CAD system can be linked to a common graphic definition of objects.

Organization strategies exploit the power of computers to construct and maintain organizations of information. Such strategies allow for quick modifications of related data. For example, Strategy 6 reminds users to make the organization of information known to the computer to (1) enhance comprehension for the user, and (2) to enable quick modifications. For example, a table constructed with tabs in a word processor is not "known" to the computer as a table (i.e. there is no internal representation of the table organization); hence the tabular structure may not be maintained when the table contents are modified. On the other hand a table, which is known to the computer as a

table, will be maintained under any modification of its contents. Similarly, data for different years in a spreadsheet can be organized in separate sheets for easy access.

Finally, visualization strategies exploit the power of computers to display information selectively without altering its content. Strategies of visualization can reduce visual overload and navigation time. For example, Strategy 9 addresses the limited screen space of most computer terminals. Often, users have tasks that require them to compare or manipulate objects that are difficult to view simultaneously on the screen. For example, a user might need to compare the contents of a table, at the beginning of a long word processing document, to the contents of a table in the middle of the same document. In such cases, instead of moving back and forth between the tables, it is more efficient to set up distinct views that focus on each table and that can be viewed simultaneously on the screen. This strategy is clearly useful in large documents containing text, numbers, or graphic elements -- and is therefore generally useful across applications using such objects.

A detailed description of the strategies shown in Figure 1, and of the efficiencies they provide, is provided elsewhere [5]. As in most performance-improvement methods, there is a trade-off between the effort needed to use a strategy and the realized gains. For example, using iteration strategies on a small number of elements may not be as compelling as using them for many elements. Nor would it be compelling to use a strategy that saves time when time is not a critical factor. Strategies are, therefore, more cost-effective for complex tasks and where the performance gains are of value to the user. Therefore it is as important to know when to use a strategy (depending on the task and context), as to know how to execute it.

Components of Strategic Knowledge

The above identification of efficient and general strategies, and their analysis through GOMS models [1, 3, 5] enabled us to identify four components of knowledge required to use strategic knowledge. (1) Users must know that there exist explicit strategies to perform particular tasks efficiently. For example, a user must know that there exists a strategy to handle exceptions when operating on a group. (2) Users must know when to use a particular strategy. In GOMS terms, there must be a selection rule that recognizes when to use this strategy. (3) Users must know how to execute a strategy. In GOMS terms, there must be a method that puts commands in the proper sequence, and operators to execute the individual commands. (4) To transfer strategic knowledge to different applications, users must know that the strategies are general, and therefore can be used to perform similar tasks in different applications. In GOMS terms, the selection rules are generally stated and can be instantiated in different task situations.

Unfortunately there are few opportunities for users to acquire all the above knowledge components. Help systems and reference manuals mainly provide knowledge of how to execute specific commands; user manuals (even those that contain advanced instruction) often provide task-specific solutions that are difficult to generalize [1, 5]; face-to-face and web-based training typically focus on teaching how to use specific commands in the context of simple tasks; and office settings rarely provide opportunities for sharing and discovering efficient methods [2].

If none of the above sources provide all the component types of strategic knowledge, how can users acquire them? The following section describes an instructional framework designed to teach these components deliberately based on an earlier attempt to teach strategic knowledge [6].

INSTRUCTIONAL DESIGN FRAMEWORK

We hypothesized that one way of ensuring that users acquire strategic knowledge is to teach it explicitly. Therefore, each of the knowledge components described earlier needs to be specifically addressed in the instructional design. To achieve this aim, we formulated the following four instructional guidelines:

1. To ensure that students know that there exist efficient strategies to perform various kinds of tasks, the instruction must provide explicit strategies and indicate the performance improvements provided by them.

2. To help students learn when to use a strategy, they must be given opportunities to explore alternative methods to perform a task, and to decide how to select an efficient method. Furthermore, students must be given opportunities to examine why they chose one method over the others, and to understand the trade-offs involved.

3. To help students learn how to execute the strategies, the students must have adequate practice to execute commands in the context of simple tasks, and must then learn to use them to execute more complex tasks with the aid of suitable strategies.

4. To enable students to transfer the strategies across different applications, students must use the strategies in the context of other applications, and recognize that they are the same strategies.

It is not sufficient just to address each of these knowledge components independently; it is also necessary to specify how to combine these components in order to construct a cohesive course. A critical issue is the order in which to teach commands and strategies. We explored the following alternatives: (1) Introduce a complex task that motivates a strategy, decompose the task into simpler subgoals, and then teach individual commands to achieve those subgoals. However, this approach failed in an early pilot study as students did not have enough command knowledge to generate alternate methods of doing the task. (2) Teach all the commands in an application, followed by using these commands together with strategies. However, this approach entails the disadvantage of excessive time elapsing between learning the commands and learning how to use them

Method of instruction	Action by instructor	Action by student
Command instruction		
1. Demonstration of commands	Demonstrate set of commands with simple tasks	Observe tasks without computer interaction
2. Practice of commands	Present practice tasks similar to tasks just demonstrated	Perform tasks using commands just learned
3. Repeat for next set of commands		
4. Summarization of commands	Review commands just taught	Observe commands being summarized
Strategy instruction		
5. Exploration of alternate methods in complex task	Present complex task which require commands just learned and which can be done in more than one way	Perform tasks independently
6. Discussion of efficiency	Ask students for alternate ways to complete task and why they chose one above the others	Discuss alternatives and rationale for choosing one
7. Demonstration of efficient method	Demonstrate task using efficient method	Observe demonstration
8. Generalization to strategy	Generalize method just used to a strategy and ask students to locate it in their hardcopy handout	Locate strategy in handout
9. Repeat for next complex task		
10. Summarization of strategies	Review strategies just taught	Observe strategies being summarized
11. Practice of strategies in similar complex tasks	Present new set of complex tasks similar to those just practiced but in a different order	Perform all tasks independently
12. Discussion of practice tasks	Ask students for strategies they used to perform each task	Discuss strategies to perform the tasks

Figure 2. The methods of instruction used to teach commands and strategies in each class.

jointly with strategies. (3) Teach a group of commands in the context of simple tasks, immediately followed by strategies that use these commands. We chose this alternative because the tight coupling between commands and strategies provides immediate practice for the commands in a different context. The tight coupling, therefore, enhances the chances that the strategy is retrieved when those commands are used.

The preceding design decisions were incorporated in a course structure illustrated by the general template shown in Figure 2. The next section will describe how this template was implemented in our experimental course.

Implementation of the Instructional Framework

The implementation of our course occurred in the context of an existing seven-week required course for freshman students at Carnegie Mellon University. The goal of this course, called the Computing Skills Workshop (CSW), is to ensure that all freshman students have basic skills to use computer applications. CSW focuses on teaching basic commands to perform simple networking tasks using UNIX®, simple word processing tasks using Microsoft® Word® (MSWord), and simple spreadsheet tasks using Microsoft® Excel® (MSExcel). To enable an experimental comparison, our implementation of the course taught the same commands, taught the same sequence of applications (UNIX, MSWord, then Excel), and took the same instruction time as the regular CSW instruction (3 classes each for UNIX, MSWord, and MSExcel, each class taking 50 minutes). The major change, as described below, was (1) in the strategic content, and (2) in the methods of instruction used to blend the general strategic knowledge with the specific commands. The two approaches could be taught in the same amount of time because the strategic content was melded tightly with command practice, and there was more efficient use of class time through the use of scripts given to the experimental instructors (as will be discussed later).

The constraints of limited time, and of teaching only a subset of commands in an application, are typical of courses offered in other universities and in software companies. Therefore, the structure of our implementation is general and potentially useful in other contexts.

At the start of each class, the students received a hardcopy handout of the strategies shown in Figure 1. This handout also contained, for the application currently taught, specific commands and examples of tasks using each strategy. Furthermore, it contained commands and examples for the applications taught in previous classes. This cumulative approach was used to emphasize that the strategies are general and useful across the applications.

Our course implementation followed the template shown in Figure 2. The first column of the template describes the methods of instruction to be used for each step, the second and third columns describe the corresponding actions to be performed by the instructors and students

The command instruction began with a demonstration of a small set of commands in the context of simple tasks (Step 1). For example, to introduce different ways to view a document in MSWord, we first demonstrated SPLIT WINDOW and SCROLL. These commands were demonstrated in the context of a three-page document that contained a table at the beginning and at the end of the document. The instructor demonstrated how to split the screen by dragging

the button located above the scroll. She then brought the tables together on the screen by splitting the screen into two panes, and then scrolling each pane to view the tables.

The students were then told to experiment with the commands just taught (Step 2). This demonstration and practice was followed by instruction for the next set of commands (Step 3). In this case these commands involved using NEW WINDOW and ZOOM. All the commands taught up to then in the class were then summarized (Step 4).

The command instruction was followed by strategy instruction. For example, the instructor opened a three-page document that had 11 different bulleted lists. The students were asked how they would move three non-adjacent bulleted items in the last list to the third list in the document. Here the instructor encouraged the students to discuss alternate methods to do the task by using the commands they had just learned (Step 5). Then the instructor stated that, by using SPLIT WINDOW or NEW WINDOW to move items from one list to another, the advantage gained was to avoid having to scroll up and down (Step 6). The instructor demonstrated this method in the practice document (Step 7). This method was then generalized to the strategy: *View parts of spread-out information to fit simultaneously on the screen.* The students were asked to locate this strategy in their handout (Step 8).

Steps 5-8 were repeated for other complex tasks demonstrating the utility of other strategies (Step 9). All the strategies presented in the class were then summarized by explicitly pointing them out in the handout (Step 10). The students were then given similar complex tasks for practice (Step 11) each of which was discussed (Step 12). The above steps were repeated for each application (UNIX, MSWord, and Excel).

The above approach contrasts with the traditional approach of teaching such applications. For example, CSW instructors are trained to teach commands in the context of simple tasks (Steps 1-3). However, the students never receive instruction of how to assemble the commands to perform complex tasks efficiently, nor do they receive any instruction on the general nature of efficient methods and thus don't acquire strategic knowledge that they could use in other applications.

ASSESSMENT OF INSTRUCTIONAL DESIGN

Our proposed instructional design is not only novel in addressing the explicit teaching of strategic knowledge, but also faces the constraints of limited time available to teach such knowledge together with commands. Hence it seemed imperative to assess the efficacy of this instructional design. Therefore, we compared our instructional design to the existing CSW course through a controlled experiment addressing the following questions:

1. Does the proposed experimental approach help the acquisition of strategic knowledge?

2. Does this approach harm the acquisition of command knowledge?

The experimental study was run in the context of the existing CSW course. Because CSW is designed to provide hands-on instruction, it is held in a laboratory with desktop computers. The course is open to technical students (those majoring in non-arts fields such as chemistry, computer science, or psychology) in the first seven weeks of the fall semester, and open solely to arts students (those majoring in fine arts, architecture, or drama) in the next seven weeks. These separate offerings of the course had been instituted by the CSW administrators because the arts students typically have much less experience with computers and thus need more assistance.

The separation enabled us to run our experimental comparison on both populations. Technical students participated in Experiment-1, and Arts students participated in Experiment-2. The main goal of these initial experiments was to compare the efficacy of the overall approaches. (Investigation of the various factors responsible for the differences between the control and experimental approaches was left to future experiments.)

Method for Experiment-1

In Experiment-1, eight of the most heavily attended CSW sections were chosen for the study. Each section contained approximately 20 students, and was balanced by student major (i.e., each section had equal numbers of students from each technical discipline). Four sections received the instruction ordinarily provided by CSW and formed the control group (with a total of 87 students). The other four sections received instruction using the experimental strategy-focused approach and formed the experimental group (with a total of 84 students). None of the students were informed that they were part of an experiment (a common practice in educational testing and approved by the Human-Subjects Clearance Committee).

Students in the experimental sections were taught all the strategies (shown in Figure 1) in the context of UNIX, MSWord, and MSExcel with the following exceptions: Strategies 4 and 5 were not taught in UNIX as the commands to execute them were too advanced for the CSW course content. Strategy 9 was not taught in MSExcel to enable us to test if students could transfer that strategy from the earlier instruction. (However, commands necessary to execute Strategy 9 were taught in MSExcel.)

Instructor Training

Each section in the course had a main instructor and a secondary instructor all of whom were undergraduate students at the university. The main instructor taught the course content in front of the classroom through a desktop computer connected to an overhead projector. The role of the secondary instructor was to provide assistance to students who had difficulty following the instruction, or had trouble with the computers. All the main instructors in the

experimental and control groups had taught CSW before, had equivalent experience in teaching CSW, and were considered to be effective instructors by the CSW administrators. All instructors received five days of training from the CSW administrators. This training focused on how to teach the commands in each application, and also provided logistical training on giving exams, grading, and pointers for effective training. Instructors were given a list of commands to teach in each application, reading material based on a commercial training guide that focused on commands, and example files to illustrate the use of commands. Each instructor practiced teaching a few commands, and received critiques from the CSW administrators.

The instructors for the experimental sections of the course received additional training for 5 days on how to teach strategies and practiced the approach with critiques. To help the instructors cope with the strategic content and the time constraints, and also to avoid possible inconsistencies between the two instructors, we gave them a script outlining what and how they were to teach the course. They were expected to use their own words and interaction style to elaborate the outline.

Post-test

At the end of the course, and after the usual CSW exams, we conducted a post-test. Students in both groups were offered $25 to perform tasks in UNIX, MSWord, and MSExcel. They were told that the tasks were to help improve the course, and that participation would not affect their grades.

The post-test tasks were designed to take a maximum of an hour and a half, and provided 13 opportunities (2 in UNIX, 5 in MSWord[1], and 6 in MSExcel) to use the 9 efficient strategies shown in Figure 1. Not all strategies in all applications could be tested because the resulting length of the post-test would then have been excessive. Therefore, there was only one strategy that was tested in all the three applications, and two that were tested in MSWord and MSExcel. The post-test tasks were different in content from the tasks taught in the experimental course, but obviously offered opportunities to use the same strategies.

In addition to attempting the tasks, the students were asked to provide handwritten descriptions of what methods they used to complete the tasks and of why they chose those methods. Interactions were recorded through a screen capture tool and command recorders. MSWord and MSExcel documents containing completed tasks were also collected.

[1] The MSWord task that attempted to test Strategy 3 failed as it did not justify using the strategy. This left 12 opportunities to use 8 strategies.

Method for Experiment-2

The method for Experiment-2 was similar to that for Experiment-1 except that the population consisted only of arts students, Furthermore, there were only two CSW sections with 24 arts students in the control group, and 25 art students in the experimental group. (The smaller number of students and sections reflect the smaller number of arts students on the campus.) The request to participate in the post-test yielded 17 from the control group, and 19 from the experimental group.

RESULTS AND DISCUSSION

Figure 3 shows the results for strategy use in the post-test for both experiments. The first column shows the strategy opportunities provided in the post-tests. For example, the strategy *Make dependencies known to the computer* (opportunity E in the figure), executed through styles, was one of the strategy opportunities provided by the MSWord post-test tasks. The numbers in the cells show the percentage of students who used each opportunity.

Effect on Recognizing and Executing Strategy Opportunities

In Experiment-1, as shown by the dark gray cells in Columns 2 and 3, the experimental group did significantly better than the control group in exploiting seven strategy opportunities ($p<0.05$ for each of the seven strategies based on chi-square tests on the frequencies in each group). These results show that students could in fact be taught to recognize opportunities to use efficient strategies, and to execute them. Furthermore, they show that this kind of strategic knowledge requires more than command instruction.

For example, each of the instructors in the control group explicitly taught how to use the *split window* command. However, only 10% of the students in that group used it in the post-test task requiring comparisons of distant cells in a large spreadsheet (strategy opportunity G). In contrast, the instructors in the experimental group taught the *split window* command to avoid scrolling and thereby illustrated the strategy *View parts of spread-out information to fit simultaneously on the screen.* As a result, 58% of the students in the experimental group used this strategy in the post-test.

Although the students in the experimental group also did significantly better than those in the control group in exploiting strategy opportunities B and H, the actual numbers of students exploiting these opportunities was small. Therefore it appears that effective teaching of these strategies would require more practice than we provided in our instruction.

In the case of five other strategies (shown in white), there was no significant difference in strategy use by students in the experimental and control groups. This result shows that some strategic knowledge can be automatically acquired just by learning commands. For example, even though the

students in the control group were given only command instruction, all of them recognized the opportunity to use formulas in the spreadsheet task (strategy opportunity I).

An analysis of these strategy opportunities revealed a possible reason why mere command instruction may suffice in some cases. For certain commands, once they are learned, the disadvantages of not using them are so great that the alternate methods pale by comparison and are possibly not even considered. For example, after having learned how to use formulas, doing manual calculations in a spreadsheet task appears ridiculous. In such cases the issue of strategic knowledge is moot as there is effectively no competition.

Another possible reason why certain opportunities were equally recognized in both groups might be due to the commands tested in the regular CSW exams. The instructors in the control group may have particularly emphasized these commands by providing more practice. Such practice, with different examples, could have helped students to learn the strategic knowledge of recognizing opportunities to use these commands. For example, both tables and formulas were tested in the CSW exams and there was no difference between the groups for strategies that used these commands.

Effect on Transfer

While the above analysis reveals whether a strategy opportunity was recognized and used, it does not reveal if the *general* form of the strategy was learned. For example, a student may know when and how to use a formula for a spreadsheet task, but may not know that formulas are just one example of setting up dependencies to be exploited later. If the strategy is not learned at the general level, it is less likely to be used in other situations involving different commands.

The question whether the students in the experimental group actually did acquire the general form of the strategy may be answered by a lengthy analysis of the self-reports provided by the students in the post-test. We are still in the process of analyzing this very rich source of data that may reveal evidence of strategic thinking. However, the current data provide some evidence that the experimental students did learn the general form of at least one strategy. As discussed before, we deliberately did not teach the strategy *View parts of spread-out information to fit simultaneously on the screen* in MSExcel in order to test whether students could transfer this strategy from MSWord. Figure 3 shows that significantly more students in the experimental group did recognize the opportunity to use *split windows* in MSExcel (strategy opportunity G) even though they were not taught the strategy in this application. While not conclusive, this result does suggest that the experimental students did acquire the general form of the strategy that enabled them to recognize its use in another application.

Effect on Command Knowledge

To check whether the added strategy content could harm the acquisition of command knowledge, we analyzed student scores in the regular CSW exams that tested mainly command knowledge. An analysis of students' mean scores revealed no statistical difference between the two groups (96.07 control, 95.54 experimental). These data therefore indicate that the experimental approach did not harm the acquisition of command knowledge.

Strategy opportunities in post-test	Experiment-1 (Tech. Stds.)		Experiment-2 (Arts Stds.)	
	Ctrl.	Exp.	Ctrl.	Exp.
UNIX				
A. Reuse and modify groups of objects	21%	79%	12%	42%
B. Check original before making copies/operating on objects	0%	13%	0%	5%
MSWord				
C. Reuse and modify groups of objects	86%	100%	59%	94%
D. Make organizations known to the computer	88%	94%	100%	100%
E. Make dependencies known to the computer	5%	62%	12%	67%
F. Exploit dependencies to generate variations	0%	46%	6%	39%
MSExcel				
G. View parts of spread-out information to fit simultaneously on the screen	10%	58%	0%	56%
H. View relevant information, do not view irrelevant information	10%	29%	18%	11%
I. Make dependencies known to the computer	100%	100%	92%	100%
J. Reuse and modify groups of objects	100%	100%	100%	100%
K. Exploit dependencies to generate variations	86%	95%	83%	93%
L. Generate new representations from existing ones	95%	98%	53%	89%

Figure 3. The percentage of students in the control and experimental groups who used the general strategies in both experiments. The dark gray cells show statistically significant differences based on chi-square tests on the frequencies in each group.

Effect on Diverse Populations

The results in Experiment-2 were similar to those in Experiment-1. As shown in Figure 3, Columns 4 and 5 show a pattern of results similar to those in Columns 2, and 3. Again, as shown by the dark gray cells, the experimental group did significantly better than the control group in six strategy opportunities. Similar to Experiment-1, the number of students exploiting strategy opportunities B and H was very small. This provides further evidence that these strategies are difficult to learn without more practice. In six

other strategy opportunities (shown in white), most users in both the experimental and control groups used the strategies with no significant difference between the two groups.

Finally, the arts students did very well (scoring an average of 95%) on the CSW test assessing command knowledge. Therefore, our instructional approach did not harm the acquisition of command knowledge. The results of Experiment-2 thus show that the course based on our instructional approach was equally useful and effective for students with appreciably different backgrounds.

SUMMARY AND FUTURE RESEARCH

To address the widespread inefficient use of complex computer applications, this paper identified a set of efficient and general strategies for using applications effectively. We then formulated a general instructional framework for teaching such strategic knowledge explicitly in combination with command knowledge. Furthermore, we implemented this framework in a university course for college freshmen students and carried out a controlled experiment in which we compared our approach to the traditional approach that predominantly focused on teaching commands.

The results of this experiment showed that our proposed approach: (1) enables students to learn efficient strategies; (2) benefits student populations with either technical or non-technical backgrounds; (3) does not require extra time compared to the traditional approach focused on command knowledge; (4) does not harm the acquisition of command knowledge; (5) has the potential of enabling the transfer of strategic knowledge across different applications.

This approach therefore provides a promising alternative to traditional training, especially because its implementation is not appreciably more complex than teaching only commands, and because it does not require excessive time. For example, our course materials have been used at the University of Western Australia to teach a similar freshman-level course with similar results [personal communication, Richard Thomas]. Furthermore, we intend to teach a similar the course in Spring 2001 at the University of Michigan.

The experiment revealed that some strategies may be automatically acquired just by learning commands. On the other hand, it also showed that other important strategies are not that easily acquired but can be learned as a result of explicit instruction.

It is possible that the efficacy of some of our instruction might be due to the fact that the instructors in the experimental group, unlike those in the control group, followed a well-designed script. Therefore, we are planning at the University of Michigan another experiment where instructors in both groups will be given well-designed scripts for their respective approaches.

In an age where computers provide a proliferation of commands and exhibit increasingly crowded interfaces, the learning of general strategies can provide users with more coherent knowledge facilitating the efficient use of complex computer applications. Furthermore, the generality of this knowledge should permit its application to new applications beyond those where the strategies were originally learned.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation, Award# IRI-9457628 and EIA-9812607. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF or the U. S. Government. The authors thank F. Peck and G. Vallabha for their contributions.

REFERENCES

1. Bhavnani, S. K. *How Architects Draw with Computers: A Cognitive Analysis of Real-World CAD Interactions*, unpublished Ph.D. dissertation, 1998, Carnegie Mellon University, Pittsburgh.
2. Bhavnani, S.K., Flemming, U., Forsythe, D.E., Garrett, J.H., Shaw, D.S., and Tsai, A. CAD Usage in an Architectural Office: From Observations to Active Assistance. *Automation in Construction* 5(1996), 243-255.
3. Bhavnani, S.K., and John, B.E. From Sufficient to Efficient Usage: An Analysis of Strategic Knowledge. *Proceedings of CHI'97* (1997), 91-98.
4. Bhavnani, S.K., and John, B.E. Delegation and Circumvention: Two Faces of Efficiency. *Proceedings of CHI'98* (1998), 273-280.
5. Bhavnani, S.K., and John, B.E. The Strategic Use of Complex Computer Applications. *Human-Computer Interaction* (in press).
6. Bhavnani, S.K., John, B.E., and Flemming, U. The Strategic Use of CAD: An Empirically Inspired, Theory-Based Course. *Proceedings of CHI'99* (1999), 42-49.
7. Card, S.K., Moran, T.P., and Newell, A. The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.
8. Cragg, P.B. and King, M. Spreadsheet Modeling Abuse: An Opportunity for OR? *Journal of the Operational Research Society* 44 (1993), 743-752.
9. Doane, S.M., Pellegrino, J.W., and Klatzky, R.L. Expertise in a Computer Operating System: Conceptualization and Performance. *Human-Computer Interaction* 5 (1990), 267-304.
10. Lee, W.O., Barnard, P.J. Precipitating Change in System Usage by Function Revelation and Problem Reformulation. *Proceedings of HCI '93* (1993), 35-47.
11. Nilsen, E., Jong H., Olson J., Biolsi, I., and Mutter, S. The Growth of Software Skill: A Longitudinal Look at Learning and Performance. *Proceedings of INTERCHI'93*. (1993), 149-156.
12. Rosson, M. Patterns of Experience in Text Editing. *Proceedings of CHI '83* (1983), 171-175.
13. Siegler, R.S., and Jenkins, E. *How Children Discover New Strategies*. Lawrence Erlbaum Associates, New Jersey, 1989.