Strategy-Based Instruction: Lessons Learned in Teaching the Effective and Efficient Use of Computer Applications

SURESH K. BHAVNANI
University of Michigan
FREDERICK A. PECK
University of Colorado
and
FREDERICK REIF
Carnegie Mellon University

Numerous studies have shown that many users do not acquire the knowledge necessary for the effective and efficient use of computer applications such as spreadsheets and Web-authoring tools. While many cognitive, cultural, and social reasons have been offered to explain this phenomenon, there have been few systematic attempts to address it. This article describes how we identified a framework to organize effective and efficient strategies to use computer applications and used an approach called strategy-based instruction to teach those strategies over five years to almost 400 students. Controlled experiments demonstrated that the instructional approach (1) enables students to learn strategies without harming command knowledge, (2) benefits students from technical and nontechnical majors, and (3) is robust across different instructional contexts and new applications. Real-world classroom experience of teaching strategy-based instruction over several instantiations has enabled the approach to be disseminated to other universities. The lessons learned throughout the process of design, implementation, evaluation, and dissemination should allow teaching a large number of users in many organizations to rapidly acquire the strategic knowledge to make more effective and efficient use of computer applications.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Curriculum, Literacy*

General Terms: Design, Experimentation, Human Factors

Additional Key Words and Phrases: Strategies, teaching, training, strategy-based instruction

This research was supported by the National Science Foundation, EIA-9812607.
Authors' addresses: S. K. Bhavnani, The School of Information, University of Michigan, MI 48109-1092; email: bhavnani@umich.edu; F.A. Peck, School of Education, University of Colorado, Boulder, CO 80309-0249; F. Reif, Carnegie Mellon University, Pittsburgh, PA 15213.

ACM Reference Format:
Bhavnani, S. K., Peck, F. A., and Reif, F. 2008. Strategy-based instruction: Lessons learned in teaching the effective and efficient use of computer applications. ACM Trans. Comput.-Hum. Interact. 15, 1, Article 2 (May 2008), 43 pages. DOI = 10.1145/1352782.1352784. http://doi.acm.org/10.1145/1352782.1352784.

1. INTRODUCTION

This article reports our experience in the design, implementation, evaluation, and dissemination of a teaching approach called *strategy-based instruction*. This approach, evolved over five years through teaching almost 400 students, is designed to teach effective and efficient strategies to use complex computer applications such as spreadsheets and Web-authoring tools.

Strategy-based instruction is motivated by several empirical studies which have reported that users have difficulty in acquiring effective and efficient strategies to use computer authoring applications. These empirical studies in the use of UNIX [Doane et al. 1990], word processors [Rosson 1983], spreadsheets [Nilsen et al. 1993; Cragg and King 1993] and computer-aided drafting (CAD) systems [Bhavnani and John 2000] have shown that, while most users can easily learn how to use basic commands, few of them acquire the knowledge to use the commands effectively and efficiently. For example, Nilsen et al. [1993], 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. However, another method of performing this task is to aggregate all the columns (including the exception), modify their widths, and then modify the exception back to its original width. This method avoids the time-consuming and error-prone steps of changing the width of each column. The method is therefore *efficient* because it reduces task time, and *effective* because it reduces errors in the end product.

While some users do in fact acquire efficient and effective methods to become experts, why do many other users persist in using inefficient and ineffective methods to perform common computer tasks? Analyses of tasks like the previous one have led researchers to conclude that users are likely to change a method to perform a task if that method fails to achieve the intended goal. However, users are more likely to not change their methods if they succeed in achieving goals, even if the methods are inefficient. For example, Singley and Anderson [1989] state:

> "productions which produce clearly inappropriate actions contribute to poor initial performance on a transfer task but are quickly weeded out. Productions that generate actions which are merely non-optimal, however, are more difficult to detect and persist for longer periods," [p. 119]

More recently, Fu and Gray [2004] suggest that most users persist in using suboptimal methods (e.g., using spaces to center a word on a page) because

they are general purpose and therefore useful for a wide range of similar tasks. Furthermore, because such suboptimal methods provide immediate incremental feedback about progress towards the user's goal, they become preferred methods over time. Unfortunately for the user, these preferred methods are highly inefficient when used for complex tasks. Other reasons (for a review see Bhavnani and John [2000]) that might conspire against users becoming more effective and efficient in using computer applications include prior knowledge dominating current performance (thus leading to the Einstellung effect [Luchins and Luchins, 1970; Flemming et al. 1997]), a production bias [Carroll and Rosson 1987] which results in users focusing on the task at hand rather than on learning to use the system more efficiently, few opportunities for acquiring effective methods in work environments [Bhavnani et al. 1996], and the lack of effective and efficient methods made explicit in instructional material [Bhavnani and John 1996; Bhavnani 1998]. Furthermore, sources of knowledge to use computer applications, like help and user manuals, either focus on command instructions for simple tasks or focus on methods to perform complex tasks methods that are task-specific and difficult to generalize [Bhavnani 1998].

Given the many reasons that conspire against users acquiring efficient and effective methods, can explicit instruction address this issue? Over the last 20 years, researchers have stressed the need for computer literacy in undergraduate education [Sellars 1988], identified the different stages of computer literacy [Halaris and Sloan 1985], and designed approaches to teach application commands such as through minimalist documentation [Carroll et al. 1987]. More recently, researchers have explored extensions of computer literacy to new uses of technology, (e.g., as a communication device) and to social aspects of technology (e.g., ethical computing guidelines) [Goldweber et al. 1994; International Society for Technology in Education 1999; Hoffman and Blake 2003]. Finally there have been numerous approaches for teaching computer skills through online tutoring systems (e.g., Shaw and Polovina, [1999]), and comparisons of different instructional approaches such as tutors and discovery learning (e.g., Charney et al. [1990]).

While this research provides important insights into the need and process of teaching users how to use computer applications, they have mostly focused on teaching how to use basic commands. However, several studies have shown that most users do not acquire efficient and effective methods just by learning how to use commands. For example, architects, despite formal CAD training to use commands and many years of experience using the CAD system, did not use effective and efficient methods in real-world tasks [Bhavnani et al. 1996]. To address this situation, we hypothesized that users might benefit from explicit instruction on effective and efficient methods to use computer applications. For example, in addition to learning how to select and modify columns in a spreadsheet, we hypothesized that users might benefit by also learning the method of dealing with exceptions. This method is general because it can be used to deal with a wide range of tasks involving different information objects (e.g., words, graphics, formulas). We refer to such general and goal-directed methods as *strategies*.

To teach efficient and effective strategies, we first need to know what they are. Unfortunately, there has been relatively little research in identifying effective and efficient strategies[1] for using computer applications. For example, as discussed earlier, Nilsen et al. [1993] identified a few efficient methods to perform spreadsheet tasks, and Lee and Barnard [1993] discuss a method to compare different parts of a document by using the SPLIT WINDOW command. In neither case has there been an attempt to generalize these methods, nor have they been organized in a framework. Furthermore, while much research has focused on teaching computer commands, we have found no attempts to teach and evaluate strategies to use computer applications.

This article describes our experience over five years to (1) develop a framework to identify and organize strategies that generalize across computer-authoring applications and (2) design, implement, evaluate, and disseminate an instructional framework to teach these strategies.

Section 2 reviews our prior research that focused on the design of the aforementioned two frameworks and how they were used to implement and evaluate a prototype for strategy-based instruction. Section 3 discusses our more recent research which evaluated the robustness of the prototype as it was extended to teach new applications to new populations and in a new context. Section 4 generalizes our experiences by indicating five lessons learned in teaching strategy-based instruction. We conclude with reflections on our experience in designing, implementing, and evaluating the strategy-based instruction to almost 400 students over five years and with research questions that need to be addressed to make more users effective and efficient in the use of computer applications.

2. DESIGN, IMPLEMENTATION, AND EVALUATION OF A STRATEGY-BASED INSTRUCTIONAL PROTOTYPE

Our research on strategy-based instruction began with the design of the following two frameworks: (1) a *strategy framework* that specified the general powers of computers which were used to identify efficient and effective strategies and (2) a *strategy-based instructional framework* that specified the general principles of instruction which were used to identify an approach for teaching strategies to novice users. These two frameworks were used to implement a prototype of strategy-based instruction, which was then evaluated in controlled classroom experiments.

2.1 Design of a Strategy Framework

The strategy framework was developed through a literature review of effective and efficient methods for using computer applications [Bhavnani and John 2000], observation of users performing real-world tasks [Bhavnani et al. 1996], analysis of the features of applications [Bhavnani and John 1998], and

[1]There have been several attempts at identifying effective strategies in other domains such as information seeking [Harter and Peters 1985], mathematics [Schoenfeld 1985], and reading and writing [Collins et al. 1989].

a GOMS [Card et al. 1983; John and Kieras 1996] analysis of key strategies [Bhavnani and John 2000]. This process led us to identify a strategy framework consisting of 9 strategies based on 4 general functions or *powers* of computer applications: iteration, propagation, organization, and visualization.[2]

The first column of Appendix-A shows the 9 general strategies, and the remaining columns show how those strategies can be instantiated across different authoring applications. For example, the strategy (described earlier) to modify many columns with an exception (an instantiation of the "handle exceptions last" strategy in Appendix-A) is efficient because it exploits the power of *iteration* provided by most authoring tools. Instead of the user modifying each column, the strategy enables the iterative task to be delegated to the computer, given some constraints. Such strategies are critical to learn because real-world users (as shown by Nilsen et al. [1993]) typically miss opportunities to use such strategies. Furthermore, a GOMS analysis of such iteration strategies found that they could lead to a reduction in time of between 40-70%, and to a reduction in the probability of errors [Bhavnani and John 2000].

Similarly, *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, the strategy "make dependencies known to the computer" is useful in word processors in the use of style. Here a user can create paragraphs that need to share a common format specification; when the specification 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 the organization of information such as tables and lists. For example, the strategy "make organizations known to the computer" is useful in a word processor in the use of a table. In contrast to using tabs to construct a table (whose organization may not be maintained when the contents are modified), using the INSERT TABLE command in MSWord enables the computer to maintain the tabular organization under any modification of its contents. Similarly, data for different years in a spreadsheet can be organized in separate sheets for easy access and manipulation.

Finally, *visualization strategies* exploit the power of computers to display information selectively without altering its content. For example, the strategy "view parts of spread-out information to fit simultaneously on the screen" addresses the limited screen space of most computer screens. For instance, 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 scrolling back and forth between the tables, it is more efficient and less error-prone to set up distinct views (e.g., through the use of the SPLIT-WINDOW command) that focus on each table

[2]We do not claim that this list is complete because we do not have a principle to generate these powers. See Bhavnani and John [2000] for a detailed description of the strategy framework.

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.

As in most performance-improvement methods, these strategies trade off the effort to use a strategy and the realized benefits. For example, iteration strategies are more beneficial when they are used for many elements rather than a few [Bhavnani and John 1998]. Furthermore, it would not be compelling to use a strategy that saves time when time is not a critical factor to a user. Therefore, strategies are more cost effective for complex tasks and where the performance gains are of value to the user. It is therefore as important to know when to use a strategy as it is to know how to execute it.

2.2 Design of a Strategy-Based Instructional Framework

As suggested by many researchers (e.g., Klahr and Carver [1988], Gong and Elkerton, [1990]), we decided to model the knowledge required to use the strategies before we designed the instruction. This approach enabled us to gain a precise understanding of the knowledge to be imparted.

A GOMS analysis of the strategies [Bhavnani and John 2000] revealed that each requires three knowledge components. (1) *Command knowledge* that includes knowledge of the existence of commands, their location, and the methods to use them. In GOMS terms, there must exist a method with operators to execute the command. (2) *Application-specific strategic knowledge* that includes knowledge of the existence of efficient strategies within an application, conditions of when to use a strategy, and the method to execute the strategy by sequencing different commands. In GOMS terms, there must be a selection rule that recognizes when to use this strategy, and an associated method to sequence different commands to execute the strategy. (3) *Application-general strategic knowledge* that provides knowledge of how particular application-specific strategies can be applied across applications. In GOMS terms, the selection rules for strategies are generally stated and can be instantiated in different task situations.

While the GOMS modeling guided us towards a more precise understanding of what to teach, it did not provide guidance on how to teach the above three knowledge components. Therefore, we exploited existing educational research to understand how to teach the knowledge components. We now describe how we designed an instructional framework by combining our understanding of the knowledge components required to use effective and efficient strategies with the existing research on how best to teach different types of skills.

2.2.1 *Command Knowledge.* Our approach of when and how to teach command knowledge was guided by previous research in the psychology and education literature. Anderson [2000, p. 387] recommended that it was important to teach component skills before teaching high-level skills that included those component skills. This suggested to us that command knowledge should be taught before strategies that used those commands. This was further verified in our early pilots [Bhavnani et al. 1999] where we attempted to first teach general strategies as a unifying framework for later teaching the commands.

However, this approach resulted in a course that was not motivating for the students because the strategies were too abstract without the command knowledge. Our final prototype, therefore, taught commands before teaching the strategies.

Prior research has also shown the importance of *active processing*, whereby students are made to engage in a task instead of merely observing passively how others perform the task. Such active processing has been shown to result in a deeper understanding of the imparted knowledge [Nicholls 1989; Nolen 1996, 2003]. We therefore designed the instruction in two parts. (1) *Demonstration of commands* where the students watched the instructor execute the steps of a command. While this first step was passive, it enabled the student to begin to acquire the declarative knowledge of the location, goal, and process of using the command. (2) *Practice of commands* where the students performed on their own a task that was different from the one demonstrated but required the same commands. Such an approach was used to encourage active processing. For example, the students were shown how to use commands to view a document such as SPLIT WINDOW, SCROLL, and NEW WINDOW, and then given an opportunity to practice the commands in the context of a new task.

Prior research also suggested that students typically have higher intrinsic motivation when they are taught with examples that are relevant to them and to the real-world [Pintrich and Schunk 1996; Myers 1989; McCade 2001; Eisenberg and Johnson 2002]. The in-class tasks for demonstration and practice (as shown in Appendix-D) were therefore carefully designed to be meaningful and relevant to the students. For example, the tasks for the technical students included organizing information related to salaries for teaching assistants, and tasks for the art students included organizing information related to art and music. These tasks were designed based on input from the student instructors who had experience in teaching the CMU freshman students in previous years, and therefore had first-hand knowledge of the tasks that were relevant to these students.

2.2.2 *Application-Specific Strategic Knowledge.* Our approach to teaching application-specific strategic knowledge was guided by research which has shown that higher retention of knowledge can often be achieved when students construct knowledge through the process of *guided discovery* (e.g., Brown and Palinscar [1989]). In addition to using the notion of guided discovery, our approach was also informed by the importance of making explicit the conditions under which a strategy is useful [Singley and Anderson 1989].

We implemented the notion of guided discovery by engaging the students in an interactive session where they were asked to describe how they would use the commands just practiced to efficiently perform a complex task. For example, after being introduced to the SPLIT WINDOW command, the students were shown a long document with many short bulleted lists and asked to discuss a method for bringing three nonadjacent items from the last list to the third list. The instructors provided feedback for the methods suggested by the students by discussing the trade-offs, and then demonstrated the efficient strategy of splitting the window before moving the items. These discussions

made explicit the conditions which best motivated the use of the application-specific strategy.

2.2.3 *Application-General Strategic Knowledge.* Our approach to teaching application-general strategic knowledge was guided by two important findings in the acquisition of knowledge: (1) the transfer of strategies can be achieved by teaching the general form of a strategy [Bossock and Holyoak 1989; Fong et al. 1986] and through multiple examples of the same strategy [Gick and Holyoak 1983], and (2) higher retention can be achieved by revisiting the same knowledge at regular and reasonably extended intervals in a phenomenon called the *spacing effect* (e.g., Hintzman [1969]; Underwood [1969], Anderson and Milson [1989]; Anderson [2000]).

We implemented the notion of teaching application-general strategic knowledge in multiple contexts by first presenting the general form of the strategy, and then showing how it could be used across many computer applications. For example, after the split-window strategy was discussed and demonstrated within an application (as discussed above), the strategy was generalized to "view parts of spread-out information to fit simultaneously on the screen" by pointing it out in a *strategy handout* (similar to the table shown in Appendix-A). This handout contained all the strategies and examples of their instantiation across the applications taught. Similarly, the application-specific strategy of using the STYLES command in Word to efficiently and effectively modify text was generalized to the strategy "make dependencies known to the computer" by pointing it out in the handout. To leverage the spacing effect to enhance retention, we taught the same strategy in subsequent applications.

2.3 Implementation of the Strategy-Based Instructional Framework

The strategy-based instructional framework was implemented as a prototype in the context of an existing seven-week required course for freshman students at Carnegie Mellon University (CMU). This course focused on teaching a set of commands to the freshman students. To provide an experimental comparison, our implementation taught the same commands, and the same sequence of applications (UNIX, MSWord, then Excel), and took the same instruction time as the regular CMU instruction (3 classes of 50 minutes each for UNIX, MSWord, and MSExcel).

The strategy-based implementation followed the template shown in Figure 1. The command instruction began with a demonstration of a small set of commands in the context of simple tasks (Step 1). For example, the instructor introduced different ways to view a document in MSWord through the use of SPLIT WINDOW and SCROLL. The students were then told to practice the commands just taught with a new task (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 until then in the class were summarized (Step 4).

The command instruction was followed by application-specific strategy instruction. For example, the instructor opened a 3-page document that had 11 different bulleted lists. The students were asked how they would move 3

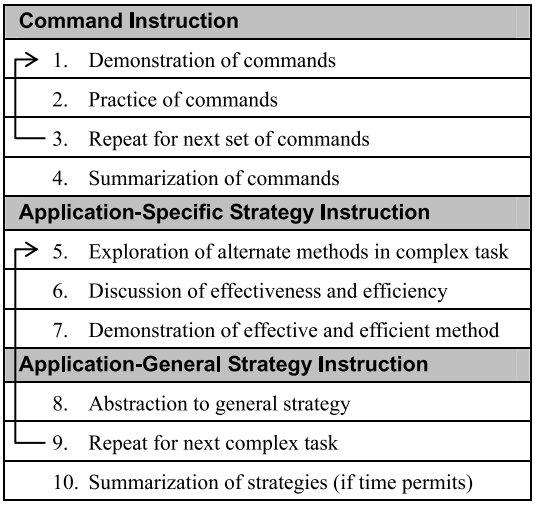


Fig. 1. Methods used to implement the strategy-based instructional framework.

nonadjacent bulleted items from 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 the advantage of using the SPLIT WINDOW or NEW WINDOW to perform the task was to avoid having to repeatedly scroll up and down between the lists (Step 6). The instructor demonstrated this method in a practice document and contrasted it with the inefficient method of scrolling (Step 7).

The students were then given the strategy handout containing the strategies and their instantiations across the applications (as previously described). The application-specific strategy just taught was abstracted to the general strategy "view parts of spread-out information to fit simultaneously on the screen". The students were asked to locate the strategy just taught in their handout and were shown how they generalized across applications (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 steps were repeated for each application (UNIX, MSWord, and Excel).

The presented approach contrasts with the traditional approach of teaching such applications. For example, instructors of the existing CMU course are taught to teach commands in the context of simple tasks (Steps 1-3). However, the students never receive instruction on how to assemble the commands to perform complex tasks effectively and efficiently. Thus they do not receive any instruction on the general nature of effective and efficient methods and do not acquire strategic knowledge that they can use across applications. Both versions of the course were taught by undergraduate students who were trained to teach the respective courses. All the time in the traditional course was spent on teaching pertinent commands and on examples illustrating their use. In the

strategy-based course, the time was spent teaching both commands and general strategies. But, because the teaching of these was tightly integrated, the total time spent by students was the same in both courses.

2.4 Evaluation of the Strategy-Based Instruction Prototype

We conducted two experiments to test the strategy-based instruction with two different populations. The first experiment (CMU-1) was conducted with science and engineering students and was designed to address the question "Does the proposed strategy-based instructional approach help the acquisition of strategic knowledge without harming the acquisition of command knowledge?"

The second experiment (CMU-2) was conducted with a population of art students and addressed the question "How effective is strategy-based instruction for teaching students with non-technical majors?"

2.4.1 *Method for CMU-1.* The students were divided into two groups. The *command group* received the instruction ordinarily provided by CMU, and the *strategy group* received the experimental strategy-based instruction. Students were randomly assigned across both treatments, and then balanced by major (i.e., each treatment had equal numbers of students from each technical discipline). This assignment resulted in 87 students in the command group and 84 students in the strategy group.

Instructor training. Each group had a main instructor and a secondary instructor, both 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. The instructors in both conditions had taught the existing CMU course before, had equivalent experience in teaching and in the use of commands, and were considered to be effective instructors. All the instructors therefore had received the same instruction on how to teach commands, but the strategy-group instructors got extra instruction to teach the general strategies. Instructors in both groups were given *teaching guides* to help teach content. The teaching guides for the command group consisted of a list of commands and practice files developed by a commercial company. This instructional approach is typically used to teach computer applications in educational and commercial organizations and therefore represented a realistic comparison condition.

The teaching guides for the strategy group included the same commands as those taught in the command group, but in addition contained instruction on how to teach the general strategies with appropriate demonstration and practice examples. Furthermore, the strategy instruction included problem-solving requiring interaction with the students. Our guides provided the overall structure for instruction but excluded the actual words to be used during instruction. Thus the guide allowed situated elaboration and improvisation by the instructors. While such teaching guides provide structure to scaffold new instructors and enable teaching consistency across instructors, they also allow for creative instructor elaboration which could lead to improved learning by

students [Bereiter 2002; Borko and Livingston 1989; Palinscar 1998; Yinger 1987]. This balance of structure and improvisation is similar to how experienced teachers typically design their instruction [Brown and Edelson 2001].

The instruction in the strategy group could be done in the same amount of time as in the command group because the strategies were tightly integrated into the teaching of the commands and concretely illustrated in the strategy handout. In addition, we created handouts to explicitly show students how the strategies in Appendix A generalized across applications.[3]

Posttest tasks. The posttest was given in a computer laboratory and consisted of three sets of tasks (one each in UNIX, MSWord, and MSExcel as shown in Appendix B). The students were presented with tasks and instructions on paper. These tasks required them to use online applications and files which consisted of a UNIX directory populated with files, MSWord to create a new file, and an MSExcel file containing a spreadsheet. The instructions also required the students to complete, after each task, a brief questionnaire in which they were asked to explain their method for completing the task and their rationale for using that method. Finally, the students were instructed to save the resulting directories and files which were checked before the students left the computer laboratory. The tasks were designed to take a maximum of 1.5 hours, but there was no time limit given to the students. The students were spaced out in the computer laboratory to ensure that they could not see the details of the computer screen of other students in the experiment.

Embedded in the previous three sets of tasks were 13 opportunities[4] (shown in the first column of Table I to use the 9 general strategies (shown in Appendix A). For example, Task 3A in MSExcel required the students to find which of two pairs of days had the lowest temperature in a large spreadsheet containing temperature data. One way to perform this task was to scroll up and down the spreadsheet in order to compare the dates. Another way to perform the task was to split the screen into two panes so that the top pane would display at all times the column headings containing the dates, and the bottom would be used to scroll through the temperature data. This approach provided a quicker and more accurate comparison of dates. Each of these strategy opportunities was different in content from the tasks taught in the experimental course.

Participation in the posttest was voluntary. Students were requested to participate in the posttest for $25 and were informed that their performance on the posttest would not affect their grade. This recruitment yielded 42 of the total 87 students from the command group, and 48 of the total 84 students from the strategy group.

[3]See Bhavnani et al. [2001] for a detailed description of the course implementation, and http://www.si.umich.edu/StrategyCourse/ for the teaching guides and handouts (also available in the ACM Digital Library).

[4]One MSWord task did not motivate the use of the strategy that it was designed to test, and therefore was excluded from the analysis. This left 12 opportunities to use 8 strategies.

Table I. Strategy Opportunities in the Posttest and Which Tasks Provided Them (Shown in Appendix B), Criteria for Using Those Opportunities, and How They Were Analyzed

Strategy opportunities		Criteria for strategy use	Method of analysis
UNIX			
A.	Reuse and modify groups of objects (Task 1)	At least one example of *mv* * in the UNIX history file (e.g., "mv *.doc docs")	Automatic analysis of UNIX history files
B.	Check original before making copies/operating on objects (Task 1)	At least one example of *ls* * in the UNIX history file (e.g., ls *.doc" used to check if theset of files to be manipulated is correct)	Automatic analysis of UNIX history files
MSWord			
C.	Reuse and modify groups of objects (Task 2A)	At least one example of *copy-paste*	Manual analysis of the screen-capture videos
D.	Make organizations known to the computer (Task 2A)	At least one example of a list created using the MSWord *list* function	Manual analysis of completed MSWord files
E.	Make dependencies known to the computer (Task 2A)	At least one style explicitly defined using MSWord *styles*	Manual analysis of completed MSWord files and screen capture video
F.	Exploit dependencies to generate variations (Task 2B)	At least one style definition explicitly modified using MSWord *styles*	Manual analysis of completed MSWord files and screen capture video
MSExcel			
G.	View parts of spread out information simultaneously on the screen (Task 3A)	At least one use of *split window* in the comparison task	Automatic analysis of MSExcel macro file, and analysis of qualitative description
H.	View relevant information, do not view irrelevant information (Task 3B)	At least one use of *zoom* in the search task	Automatic analysis of MSExcel macro file, and analysis of qualitative description
I.	Make dependencies known to the computer (Task 3C)	At least one formula used	Manual analysis of completed MSExcel files
J.	Reuse and modify groups of objects (Task 3C)	At least one example of *copy-paste* or *fill-bar*	Automatic analysis of MSExcel macro file
K.	Exploit dependencies to generate variations (Task 3D)	At least one example of a dependent cells modified	Manual analysis of completed MSExcel files
L.	Generate new representations from existing representations (Task 3E)	At least one use of Charts	Manual analysis of completed MSExcel files

2.4.2 *Method for CMU-2.* The method for CMU-2 was similar to CMU-1 except that the population consisted of only art students. The command group and the strategy group consisted of 24 and 25 art students, respectively. Similar to the CMU-1 experiment, students were requested to participate in the posttest for $25. The recruitment yielded 17 students from the command group and 19 students from the strategy group.

2.4.3 *Data Collection.* We collected and analyzed five types of data.

(1) *Screen-capture videos*, which recorded the computer interaction of each student.
(2) *Command logs*, which consisted of UNIX history files and MSExcel Macro files. These files contained a list of commands executed by the students in a format that could be automatically analyzed by computer scripts.
(3) *Completed task files*, which were collected in Word and Excel.
(4) *Qualitative descriptions,* authored by each of the students, in which they explained how they completed each task, and the rationale for their method were collected.
(5) *Exam scores*, which were based on an exam required by all students enrolled in the course and tested only command knowledge were used to check whether the strategy instruction harmed command knowledge.

2.4.4 *Analysis.* The focus of our study was to analyze whether or not students recognized the opportunity to use a strategy. Therefore, for each student, we analyzed each of the 12 strategy opportunities for evidence of strategy use. For each opportunity, students were given a binary score indicating whether or not they used the strategy in that particular opportunity. This led to nominal-level data for each strategy (i.e., a student either used a strategy or did not). Table I shows the strategy opportunities for each task, the criteria for strategy use, and the method used to analyze the strategy.

Where possible, we used computational methods to analyze the data in order to reduce errors. In all cases, whenever there was any chance for ambiguity, the data were double-checked in another form such as the screen capture video or the written descriptions. For example, as shown in row E of Table I, Task 2A in MSWord included an opportunity to use the strategy "make dependencies known to the computer". Students were given credit for having used this strategy if they explicitly defined and used at least one style using the MSWord STYLE command. Strategy use was assessed by first looking at each student's completed task file for evidence of style use, and then confirmed through analysis of the screen capture videos. This confirmation was necessary because MSWord sometimes automatically assigns styles to text.

2.4.5 *Results: CMU-1.* We performed a two-step analysis of the data. First, we tested for an overall effect by analyzing the proportion of strategy opportunities used by each student in the command and strategy groups. For the command group, the mean proportion of strategies used was 42.66% [SD=.16], while for the strategy group, the mean proportion of strategies used was 68.58% [SD=.21]. Therefore, a typical student in the command group used 42.66% of the strategy opportunities shown in Table II, while a typical student in the strategy group used 68.58%. This difference was statistically significant based on a t-test of the proportions ($t(88)=6.73$, $p<.001$).

In the second step of our analysis, we tested each strategy individually to provide a more fine-grained view of the data. As reported elsewhere [Bhavnani et al. 2001] and as shown in Table II, the strategy group did significantly better

Table II. Results from the CMU-1 and CMU-2 Experiments (Grayed cells show statistically significant differences between the command and strategy group (based on a Chi-square test at the .05 level). See Table V in Appendix E for details of each statistical comparison.)

Strategy opportunities		**CMU-1**		**CMU-2**	
		Com.	**Strat.**	**Com.**	**Strat.**
UNIX					
A.	Reuse and modify groups of objects (*mv* *)	21%	79%	12%	42%
B.	Check original before making copies/operating on objects (*ls* *)	0%	13%	0%	5%
MSWord					
C.	Reuse and modify groups of objects (*copy/paste*)	86%	100%	59%	94%
D.	Make organizations known to the computer (*lists*)	88%	94%	100%	100%
E.	Make dependencies known to the computer (*styles*)	5%	62%	12%	67%
F.	Exploit dependencies to generate variations (*modify styles*)	0%	46%	6%	39%
MSExcel					
G.	View parts of spread out information simultaneously on the screen (*split window*)	10%	58%	0%	56%
H.	View relevant information, do not view irrelevant information (zoom)	10%	29%	18%	11%
I.	Make dependencies known to the computer (*formulas*)	100%	100%	92%	100%
J.	Reuse and modify groups of objects (*drag across cells*)	100%	100%	100%	100%
K.	Exploit dependencies to generate variations (*modify dependent cells*)	86%	95%	83%	93%
L.	Generate new representations from existing representations (*charts*)	95%	98%	53%	89%

than the command 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[5]). Furthermore, there was no statistically significant difference in command knowledge between the two groups as measured by mean scores on the in-class exams which tested only command knowledge as discussed previously (mean for command group: 96.07, mean for strategy group: 95.54 $t(511)=0.63$, $p=.53$).

The results demonstrate that students could, in fact, be taught to recognize opportunities to use efficient strategies and to execute them. Closer inspection showed that five strategies opportunities (D, I, J, K, L in Table II) may be easily acquired just by learning commands. For example, even though the command group was given only command instruction, all of them recognized the opportunity to use formulas in the spreadsheet task (I). One explanation is that once commands such as formulas in MSExcel are learned, the alternate methods (e.g., doing manual calculations in the spreadsheet) are just too inefficient to be considered. This could also explain why two sets of strategy opportunities (F/K and C/J), each testing the same general strategy but with different commands, had very different usage profiles especially for the command group. These results also confirm laboratory studies which show that, under certain conditions, a strategy to reuse information through cut-and-paste can be discovered just by knowing commands [Charman and Howes 2003]. Future

[5]Chi-square tests were used because the data were nominal as described in Section 2.2.4. See Table V in Appendix E for details of each statistical comparison.

research needs to explore more closely what makes certain strategy opportunities more difficult to detect compared to others.

Although the strategy group did significantly better for strategy opportunities B and H, the actual numbers of students exploiting those opportunities was small. This suggests that these strategies required more instruction than we provided. However, because zoom often leads users to become disoriented, the lack of zoom use could also be a conscious choice to avoid such problems. Overall, the results showed that most of the strategies can be taught in the same amount of time as the regular approach without harming the acquisition of command-knowledge.

CMU-2. As shown in Table II, the results of CMU-2 were similar to those in CMU-1. The overall effect remained (command group mean proportion: 37.25% [SD=.19], strategy group mean proportion: 58.33% [SD=.20], $t(34)=3.21$, $p<.01$). Furthermore, six of the strategy opportunities showed a significant difference between the two groups. These results show the utility of the strategy-based instruction for students with very little technical background. Finally, command knowledge was significantly higher in the strategy group than the command group (command group mean: 86.71; strategy group mean: 95.47, $t(151)=3.97$, $p<.001$). This suggests that teaching commands with strategies might have the added benefit of improving command knowledge.

2.5 Summary of the Strategy-Based Instructional Prototype

In Phase-1 of our research we (1) identified a strategy framework that helped to organize nine effective and efficient strategies that were general across applications, (2) identified a framework to teach the knowledge components, (3) implemented the instructional framework, and (4) tested the instructional design in two controlled experiments. Furthermore, we learned that a few strategies did not require explicit instruction, while others required more instruction than we expected.

The results of these experiments showed that the strategy-based instruction (1) enabled students to learn effective and efficient strategies, (2) benefited student populations with both technical and nontechnical majors, (3) did not require extra time compared to the traditional approach focused on command knowledge, and (4) did not harm the acquisition of command knowledge.

In our next phase, we tested whether the instructional framework could be extended to new applications and in a different context compared to the prototype.

3. EXTENSION TO A NEW APPLICATION AND A NEW POPULATION

Because the first author moved to the University of Michigan, we had the opportunity to test whether the strategy framework and the instructional framework were robust in the new university context with a different population of freshman art students. Such a test of robustness is critical because educational interventions can easily fail to produce positive results when used in a context where the original authors have less control [Brown 1992].

However, testing the robustness of an instructional approach in a new real-world context usually comes at the cost of trading off experimental control, a common problem in testing educational interventions in real world contexts [National Research Council 2000; Brown 1992]. As described in the following section, we had to make many modifications to the instructional design to fit the new context. This loss of control prevented us from making statistical comparisons with the experiments at CMU. Furthermore, unlike many user tests of systems that take around an hour per user, educational field experiments may take an entire semester. This long time span reduces the kind of manipulations that can be done practically. However, the experience of testing our approach in a new context led to insights of using strategy-based instruction in new contexts and how the approach could be disseminated to different institutions.

3.1 Extension of the Strategy-Based Instructional Design

The new context had three major differences that tested the robustness of our strategy and instructional frameworks. (1) UNIX was not taught as it was not considered an application that was particularly useful for the Michigan Art students. (2) The Art Department faculty requested that the course be extended to teach Dreamweaver (a Web-authoring application). (3) At the request of the faculty, an extra day was added for teaching each application to enable the students to perform a summative task (e.g., creating a resume in Word). To test the changes, we asked the question: "Could the strategy-based instructional approach be applied to teach new applications?"

3.1.1 *Extension to a New Application.* To develop teaching guides for Dreamweaver, we followed a three-step process. (1) Identify in each application the commands that were appropriate to teach freshman art students. (This was done in consultation with the Arts faculty.) (2) Instantiate the 9 general strategies in Dreamweaver. (3) Construct for each application a 4-day teaching guide that closely followed our original instructional design framework but added an extra day for each application.

We decided to teach in Dreamweaver 17 commands, which the Arts faculty agreed were useful for students, to build a basic Web site where the students could upload their graphics and music files. The commands ranged from OPEN NEW FILE to DEFINE WEBSITE and PUBLISH WEBSITE. Appendix A shows how each of the 9 strategies were instantiated in Dreamweaver using the chosen commands. For example, the strategy "make organizations known to the computer" was instantiated in Dreamweaver through commands to create and modify tables which are used by most Web site designers to organize content in a Web page. Appendix D shows the portions of the Dreamweaver teaching guide concerned with the commands to use tables.

While the strategies themselves generalized with minimal effort, our main difficulty was in designing the demonstration, practice, and problems-solving tasks for Dreamweaver. This was because, while it is easy to construct Web sites that have minimal functionality, a credible looking Web site that was motivating and relevant to the Art students required a considerable effort in

graphic design. As we did not have these graphic design skills, we employed a graphic designer to construct a Web site with many well-designed pages, each of which demonstrated the use of the commands and strategies that we had chosen to teach.

Therefore, we were able to instantiate the nine strategies in a new application and to create a teaching guide for the new application. This demonstrated that we were successful in extending both the strategy framework and the instructional design framework to a new application. Furthermore, we learned that applications varied in the amount of setup costs to build demonstration and practice examples that were relevant and motivating.

3.2 Effect of Strategy-Based Instruction on a New Population

To test the robustness of the instructional design in a new context, we conducted an experiment (henceforth referred to as UM-1) with the Art students at Michigan. Our goal led to the following research question: "How robust was the strategy-based instructional approach when implemented in a new context with a new population, and with less control?"

Because retention of knowledge is critical in learning, we also wished to investigate how well the strategies were retained over time. This was not investigated in the CMU experiments as the posttests were given immediately after the instruction set was complete. The above goal led to the following research question: "Does the strategy-based instructional approach help the retention of strategic knowledge over time?"

In addition to the primary research goals to test robustness and retention, we also used this opportunity to perform a small exploratory study to probe the role of the general form of a strategy in its transfer across applications.

3.2.1 *Method for Experiment with Art Students (UM-1).* As discussed in Section 2.2, our GOMS modeling helped to pinpoint three types of knowledge that were important for the strategic use of computer applications: (1) Command knowledge (e.g., the existence of the split window command, its location, and the method of how to use it); (2) Application-specific strategic knowledge (e.g., the existence of a strategy to modify distant parts of a MSWord document by using the split-window command); (3) Application-general strategic knowledge (e.g., the general form of the application-specific strategy so that it can be used across applications). The CMU-1 and CMU-2 experiments compared students who were taught only command knowledge (command group), to students who were taught all the three knowledge components (strategy group).

In the current experiment, fifty Art students at the University of Michigan were randomly divided into two equal groups and then balanced by prior experience in Word and Excel. One group was given the same instruction as the strategy group in the CMU experiments (with the modifications discussed in Section 3.1) and is therefore still called the strategy group. Analysis of how the strategy group performed the posttest tasks will reveal robustness of the instructional framework when taught in a new context.

In addition to this analysis, we explored the explicit role of the general form of the strategy by teaching the second half of the class only command

Table III. Knowledge Components Taught to the Ap-Specific and the Strategy Groups in UM-1.

Knowledge components taught	**Command + Ap-specific**	**Strategy**
Command knowledge	Yes	Yes
Application-specific strategies	Yes	Yes
Application-general strategies	No	Yes, except the strategy *view parts of spread out information simultaneously on the screen* in Excel

knowledge and application-specific strategic knowledge. For example, while we taught how to use the SPLIT WINDOW command and when to use it strategically within MSWord, we did not teach the general form of the strategy "view parts of spread out information simultaneously on the screen". Because this group was taught command knowledge and only the application-specific strategic knowledge, it was called the command + ap-specific group.

Furthermore, to test if the general form of the strategy was necessary for transfer across applications, we also did not teach the strategy "view parts of spread out information simultaneously on the screen" in Excel in the strategy group. Therefore, while the command + ap-specific group had no general strategy instruction at all, the strategy group had instruction for all general strategies in all applications except for the above strategy in Excel. This manipulation allowed us to explore if (1) the general form of the strategy was necessary for its use within an application, and (2) if the general form was necessary for transferring that knowledge across applications. The latter was tested in a single condition for exploratory purposes. Table III shows the different knowledge components taught in each group.

The posttest data were collected as part of the final exam required by all students. This yielded 25 students in each condition. The posttest for each group was identical to the MSWord and MSExcel posttest tasks used in CMU-1. Because tasks for Dreamweaver require long set-up times exceeding the time constraints of the exam, we were unable to test strategies in that application.

The retention test (see Appendix C) was conducted one month later to explore how well the students in both conditions retained the strategies. The students were requested to take part in the retention study for \$40[6]. This yielded 18 and 20 students from the command + ap-specific group and the strategy group, respectively, where all these students constituted 76% of the original class. There was no statistically significant difference between the posttest scores of the students taking the retention test and the posttest scores for the students who did not take it. This result suggests the absence of a self-selection bias in the students who opted to take the retention test. The students in the retention test were given isomorphs of the posttest tasks described earlier, and the analysis criteria were identical to those used in the CMU experiments.

[6]This incentive was determined appropriate given that the course had already been completed.

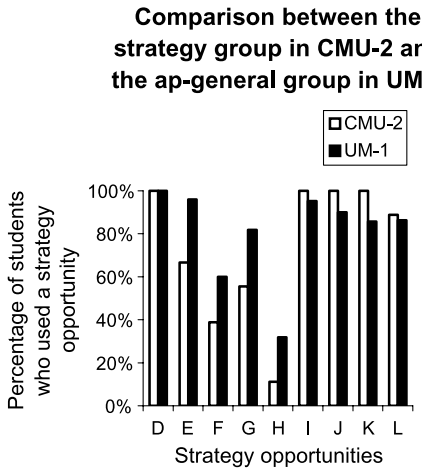


Fig. 2. Comparison between the use of strategy opportunities in the strategy group in CMU-2 and the strategy group in UM-1. While a statistical comparison is not possible because of the extra day on each application, the use of strategy opportunities in UM-1 is higher in 4 strategy opportunities and only slightly lower in the rest. Full names of each strategy opportunity are shown in Table IV.

3.2.2 *Results and Discussion.* As discussed earlier, a statistical comparison between UM-1 and CMU-2 could not be done because the UM-1 students had one extra day of instruction per application. We therefore present the data in Figure 2, mainly to provide a direct comparison with the CMU-2 data. As shown[7], the strategy group had a high rate of strategy use, comparable to the strategy group in CMU-2 (also Art students). In four strategy opportunities (I, J, K, and L), there was a small drop (5%, 10%, 7%, and 11%) in strategy use. The rest had either equal or much more strategy use. The extra instruction might explain the higher scores that UM-1 students acquired in the posttest for strategy opportunities. But the results certainly suggest that the instructional framework was robust in a new context.

Just as in the CMU experiments, we performed a two-step analysis procedure. In the first step, we tested for an overall effect by analyzing the proportion of strategy opportunities used by each student in the command+app-specific and in the strategy groups. This test showed that the mean proportions for both groups was very high and that no significant difference existed between the two groups (command+ap-specific group mean proportion: 72.89% [SD=.18], strategy group mean proportion: 72.44% [SD=.24], $t(48)=.0034$, $p=.997$). This result was expected because, with the exception of "view parts of spread out information simultaneously on the screen" in MSExcel, both groups were taught application-specific strategies in the applications in which they were assessed.

As discussed earlier, neither the command + app-specific group nor the strategy group in UM-1 was taught the general form of the strategy "view parts

[7]Strategy opportunity C (reuse and modify groups of objects copy/paste) could not be analyzed because of corrupted screen capture data.

Table IV. Results from the UM-1 Experiment. Grayed cells show statistically significant differences between the command + ap-specific and the strategy groups (based on a Chi-square test at the .05 level). The single statistically significant result between the two groups suggests the importance of teaching the general form of the strategy to achieve transfer across applications (which was tested in only one case). The rest of the nontransfer situations show no difference between the two conditions shown, and between these and the CMU strategy group (as shown in Figure 2), suggesting that the strategy-based instructional approach was successful in a new context. Strategy opportunity L was designed as an extra-credit question. See Table VI in Appendix E for details of each statistical comparison.

Strategy Opportunities		**UM-1 (Post-test)**		
		Command + Ap-specific	**Strategy**	**Explicitly Taught**
MSWord				
D.	Make organizations known to the computer (*lists*)	100%	100%	Yes
E.	Make dependencies known to the computer (*styles*)	92%	96%	Yes
F.	Exploit dependencies to generate variations (*modify styles*)	52%	60%	Yes
MSExcel				
G.	View parts of spread out information simultaneously on the screen (*split window*)	54%	82%	No
H.	View relevant information, do not view irrelevant information (*zoom*)	38%	32%	Yes
I.	Make dependencies known to the computer (*formulae*)	100%	95%	Yes
J.	Reuse and modify groups of objects (*Drag across cells*)	78%	90%	Yes
K.	Exploit dependencies to generate variations (*modify dependent cells*)	74%	86%	Yes
L.	Generate new representations from existing representations (*charts*)	92%	86%	Yes

of spread out information simultaneously on the screen" in MSExcel. However, the strategy group was taught the general form of the strategy in MSWord, while the command + ap-specific group was taught only the application-specific form of the above strategy in MSWord. To test if this had an effect on transfer, we performed the second step of our analysis, a Chi-square test on each individual strategy opportunity. As shown in Table IV, there were no significant differences between the groups on directly-instructed strategies (see Table VI in Appendix E for details of each statistical comparison). However, based on a Chi-square test on the frequencies in each group, there was a significant difference ($df=1$, $n=50$) = 3.99, $p<.05$) between the command + ap specific group and the strategy group for the untaught, transfer strategy (G) "view parts of spread out information simultaneously on the screen" in MSExcel. This result suggests that transfer is improved when a strategy is taught in its general form. This result is, however, not definitive because we tested for transfer in only one instance. However, despite the fact that the split window command (necessary to execute the strategy in Word and Excel) is identical in both applications, the results indicate that application-general strategic knowledge may be important to enable transfer.

Transfer has been difficult to achieve in many studies (see Singley and Anderson [1989] for an extensive discussion). It is possible that transfer did occur

in our exploratory study because we (1) explicitly taught the conditions of when to use the strategy in the context of tasks in a specific application, (2) made users aware of the nature of tasks that warrants the strategy, and (3) taught the strategy multiple times with different examples. Because this was true for all the strategies that we taught, we believe our instructional framework appears to be well-suited for the transfer of strategies and is in agreement with earlier research on transfer of skills [Bossock and Holyoak 1989; Fong et al. 1986; Gick and Holyoak 1983]. However, future research should test this result with more extensive transfer conditions.

Finally, the results of the retention test showed high retention of strategic knowledge across all the strategy opportunities. None of the differences between the posttest and retention test were significant in MSExcel or MSWord for either group. Thus, retention of strategic knowledge after one month was high in both groups.

3.2.3 *Dissemination of the Strategy-Based Instructional Approach.* The success of our strategy-based instructional approach in two universities led to requests for the use of the course material by the School of Nursing at the University of Michigan. The Nursing faculty requested that the course be used to teach their freshman students the strategic use of computer applications. They also requested that we teach PowerPoint in addition to MSWord, MSExcel, and Dreamweaver. The course was taught in two iterations at the Nursing School with minimal involvement of the original authors. This was achieved by providing written instructions to graduate students who were hired to teach the course. These instructions included the teaching guides (discussed in Section 2.4.1) and a set of guidelines[8] of how to design and execute the teaching guides. The graduate student instructors modified and executed the course with minimal involvement of the original authors. Additionally, we received requests from two other universities[9] that wished to explore how to provide strategy-based instruction to freshman students.

3.3 Summary of Extending Strategy-Based Instruction to New Applications and Populations

In Phase 2 of our research on strategy-based instruction, we tested whether the strategy and instructional frameworks could be extended to a context requiring a new computer application and using a new population of Arts students. Furthermore, we tested whether strategic knowledge could be retained over time. The results showed that the strategy framework and the instructional framework could be successfully extended to new applications and to new populations and that strategic knowledge was retained even after one month. Furthermore, an exploratory study suggested that transfer of strategic skill to a

[8]These guidelines are available from http://www.si.umich.edu/StrategyCourse/.

[9]See Thomas and Foster [2001] and Marsh [2007] for a description of how the course was implemented and evaluated at the University of Western Australia, and at the Walter Sisulu University, South Africa, respectively. These experiments further demonstrate the robustness of our approach when implemented in new contexts and with new populations.

new application improves significantly when the strategy is taught in its general form. Finally, the course material was disseminated to another context within the University of Michigan and to the University of Western Australia.

4. LESSONS LEARNED IN TEACHING THE STRATEGIC USE OF COMPLEX COMPUTER APPLICATIONS

Over the five-year span of our research program, we have learned lessons related to (1) the need to teach strategies explicitly, (2) the organization of strategies, (3) approaches to teach commands and strategies, (4) guidelines for creating and teaching strategy-based instruction, and (5) the effects of strategy-based instruction on learning.

4.1 Strategies Need To Be Taught Explicitly

Several studies have shown that many users do not acquire effective and efficient strategies to use computer applications. In fact, as discussed in Section 1, there are many reasons that conspire against users discovering and using helpful strategies, despite many years of experience and despite well-designed interfaces. While we have experimented with online intelligent help systems [Bhavnani et al. 1996] and are open to other approaches to deliver instruction, we have come to believe that whatever the medium and style of instruction, most users need to be taught strategic knowledge explicitly before they acquire a wide range of effective and efficient strategies.

4.2 Strategies Exploit General Powers of Computers

To teach strategies explicitly, we first need to know what they are. While several studies had identified the need to teach effective and efficient strategies, none of the studies had identified a systematic approach to organize them. Because we wished to identify strategies that generalized across authoring applications, we focused on the general functions or powers that these applications provided. This led us to organize strategies based on four powers of computer applications: iteration, propagation, organization, and visualization. In addition to helping us to organize the general strategies and instantiate them systematically across applications, the framework can be extended to include new strategies as new powers of computer applications are discovered and provided to users.

4.3 Strategies Should Be Taught in Combination with Commands

A critical component of strategy-based instruction is that commands should be tightly integrated with the teaching of strategies. Through our early prototypes, we learned that an effective way to teach strategies was to first teach how to use a small set of commands, then immediately teach when to use those commands, followed by teaching the general form of the strategy. Other

approaches, such as providing a general framework of all the strategies before teaching commands, did not motivate students in the classroom. We believe this is because learning strategies, in the absence of knowing how to implement them with commands, is too abstract.

4.4 General Strategies Can Enable Users to Acquire Strategic Knowledge

While there has been research in teaching strategies to perform tasks in a wide range of domains such as math and reading, we found no systematic studies that explored how to teach general strategies to use computer applications. Our research has shown that, for the most part, merely learning commands does not easily lead users to acquire many strategies. Our experiments have shown that to learn how to use efficient and effective strategies to use computer applications in a short amount of time, users should be explicitly taught (1) the commands needed for a strategy, (2) the conditions under which a strategy is useful, and (3) the strategy itself in its application-specific form. If users are expected to transfer strategies across applications, there is some indication that the general form can significantly improve transfer even when the commands to use the strategies are virtually identical.

5. REFLECTIONS

This article has focused on our five years of research related to strategy-based instruction. However, our research path began much earlier when we first conducted an ethnographic study to observe how architects were using CAD systems to perform real-world tasks [Bhavnani et al. 1996]. This study revealed that the architects were not using effective and efficient strategies despite knowing how to use the commands on the interface and despite many years of experience in using the CAD system. Cognitive analysis of these real-world tasks suggested the existence of strategies that could improve the performance of the users. Because the observed users had few problems with the interface, we decided to focus on strategy-based instruction to address the ineffective and inefficient use of computer applications.

Our decision to pursue strategy-based instruction has often been criticized for attempting to change users to fit poorly designed systems. This argument takes the position that the need for instruction represents a failure in the design of the interface and therefore, that instead of attempting to change the user through instruction, we should attempt to change the interface so that it enables users to spontaneously discover and use effective strategies.

We believe this argument ignores important characteristics of the problem. Consider the SPLIT WINDOW command available in MSWord and MSExcel. This command is explicitly designed to perform the simple goal of dividing a window into two panes and is useful in a wide range of higher-level editing tasks. However, while a user might learn from the interface that the split window command can divide the window into two panes, it is much more difficult

to learn from the same interface when best to use that command. To know when to divide the window into two panes, a user must recognize specific characteristics in the higher-level task. For example, a user must learn to recognize that when two information objects that need to be compared are far apart in a document, then they need to be brought together on the screen before performing the comparison[10]. Acquiring knowledge to recognize such task-related cues is difficult, and as demonstrated by our experiments, such recognition often does not happen spontaneously just from knowing how to use commands. As described in Section 2, knowledge to detect characteristics of a higher-level task and connect them to specific commands, has been a critical part of the strategic knowledge that we have abstracted and taught.

One can argue that interfaces could automatically detect characteristics of a task from user actions and then suggest to the user when to use more efficient and effective methods. This approach can be successful when the detection is unambiguous (like inferring that a user is manually creating a numbered list and automatically converting the text into a numbered list). However, in most cases, automatic detection of task characteristics is not unambiguous. For instance, even intelligent interfaces would have difficulty in unambiguously inferring that a user was attempting to compare two distant pieces of information and correctly suggest the use of the split window command. Such ambiguity can lead to complex [Carroll and Aaronson 1988] and often annoying dialogs with the user to resolve the ambiguity.

We have explored approaches to automatically detect opportunities for using more effective strategies [Bhavnani et al. 1996], and believe such research should continue and be informed by the results reported here. For example, such projects should target those strategies, like the ones identified in our experiments, which are difficult to acquire just by knowing commands. However, we believe that such attempts should, where possible, be complemented with strategy-based instruction that can assist users in acquiring a comprehensive understanding of the power of computers and how best to exploit them. Such knowledge, as we have explored, could have the added advantage of being transferable across applications.

Looking back, our research has attempted to reduce the cost of learning by using strategies. This was achieved by teaching users how to use commands, how to recognize opportunities to use them strategically within an application, and then how such opportunities generalize across applications. While our overall approach has been fairly successful, it may be insufficiently motivating for students who already have a substantial knowledge of commands. Accordingly, we believe that it might be useful to develop a course that focuses only on strategic knowledge without also teaching commands. We are also exploring the development of a minimalist strategy manual that would provide brief online instruction of strategies for use with different computer applications. Furthermore, while we have focused on delivering strategy instruction in a

[10]See Flemming et al. [1997] for a discussion on how characteristics in high-level architectural drawings tasks need to be recognized in order for users to make more effective and efficient use of CAD systems.

classroom context, we would also like to explore how the same material could be delivered through computer-based tutors.

Future research also needs to explore how well the strategies transfer across applications and are retained over longer periods of time. Furthermore, as discussed by others [Payne et al. 2001], we need to better understand the attributes of and conditions under which some strategies are automatically acquired just by learning commands. More research is needed to investigate how best to teach instructors how to design and execute strategy-based instruction effectively.

Finally, the ineffective use of computer applications is not unique to authoring applications. Many users have difficulty in acquiring strategies to perform effective searches on the Web [Bhavnani 2001], and we believe that our instructional framework could be adapted to teach strategic knowledge to improve information-seeking behavior [Bhavnani 2005; Bhavnani et al. 2006]. Our hope is that such research will help achieve our ultimate goal of making users more effective and efficient in the use of a wide range of computer applications.

APPENDIX. A NINE GENERAL STRATEGIES AND THEIR INSTANTIATIONS ACROSS AUTHORING APPLICATIONS

General Strategies	Instantiation in MSWord	Instantiation in MSExcel	Instantiation in Dreamweaver	Instantiation in MSPowerPoint
Iteration Strategies				
1. Reuse and modify groups of objects	Copy and modify existing paragraph to create a new one	Copy and modify existing data sets and formulae to create new ones	Reuse and modify existing Web page	Copy and modify existing slide to create a new one
2. Check original before before making copies/ operating on objects	Check if paragraph is correct and complete before making copies	Check if information is correct and complete before copying to create new tables	Check navigation set of links before copying to other pages	Check if slide text/format is correct and complete before making copies
3. Handle exceptions last	Modify full paragraph, then modify exception	Modify entire table to one format, then modify exception to another	Copy set of links to another page, then unlink exception	Modify entire slide text, then modify exception
Propagation Strategies				
4. Make dependencies known to the computer	Setup paragraph formats to be dependent on styles	Setup formulas to be dependent on data	Define a site so that links are dependent on the location and names of the files they point to	Use a slide master to automatically provide a consistent look to every slide in a presentation
5. Exploit dependencies to generate variations	Modify style definitions to generate variations of the same document	Modify data to generate different results for the same data set	Modify location and names of files in a site to generate variations of site structure	Modify the slide master to generate variations of the same slide presentation
Organization Strategies				
6. Make organizations known to the computer	Use lists and tables to organize related information	Use worksheets to organize data sets	Use lists and tables to organize information in a Web site	Use slide layout to organize each slide
7. Generate new representations from existing ones	Generate table from tabbed words	Generate chart from table	Generate Web version of page in browser	Generate numbered list from bulleted list
Visualization Strategies				
8. View relevant information, do not view irrelevant information	Zoom in to reveal more detail	Zoom out to locate cells	Un-display markers	View all the slides in a presentation in slide sorter view
9. View parts of spread-out information to fit simultaneously on the screen	Use split windows or new windows to bring relevant information simulta- on the screen	Use split windows or new windows to bring relevant information simulta- on the screen	Use a site map to visualize relationships between pages	Use new windows to bring relevant information simulta- neously on the screen

APPENDIX B. POSTTEST TASKS

Task 1: UNIX

Step 1: Open Nifty Telnet (Start ➔ Communications ➔ Nifty Telnet)

Step 2: Type the following command:

```
source ~fap/startup
```

A directory named **`project`** has been created in your current directory.

Please raise your hand if you have any problems.

Task 1: In the **`project`** directory, organize the files using directories so that:

1. The files are easy to display and
2. Any specific file is easy to retrieve.

Note: The project directory contained the following files:

```
chem-lab-01.doc
chem-lab-01.xls
chem-lab-02.doc
chem-lab-02.xls
chem-lab-03.xls
chem-lab-04.doc
data1-stats.xls
data2-stats.xls
final-draft-hist.doc
first-draft-hist.doc
hmwk1-math.xls
hmwk2-math.xls
hmwk3-math.xls
hmwk4-math.xls
hmwk5-math.xls
hmwk6-math.xls
hwwk1-stats.doc
hwwk2-stats.doc
hwwk3-stats.doc
hwwk4-stats.doc
new-resume.doc
resume.doc
rewrite-hist.doc
second-draft-hist.doc
second-resume.doc
```

Task 2: MS Word

Task 2A: Please use MS Word to create the document below (Fonts need not be exactly the same as those shown). Design the document so that its appearance is easy to modify.

Agenda for United States Senate, 1 October

Bureau of Transportation

Bill TR-987

- Overview of United States Senate Bill TR-987
- Presentation of United States Senate Bill TR-987
- Vote on United States Senate Bill TR-987

Bill TR-236

- Overview of United States Senate Bill TR-236
- Presentation of United States Senate Bill TR-236
- Vote on United States Senate Bill TR-236
- Closing Remarks

Bureau of Housing

Bill HS-345

- Overview of United States Senate Bill HS-345
- Presentation of United States Senate Bill HS-345
- Vote on United States Senate Bill HS-345

Bill HS-632

- Overview of United States Senate Bill HS-632
- Presentation of United States Senate Bill HS-632
- Vote on United States Senate Bill HS-632
- Closing Remarks

Bureau of Urban Development

Bill UD-696

- Overview of United States Senate Bill UD-696
- Presentation of United States Senate Bill UD-696
- Vote on United States Senate Bill UD-696

Bill UD-750

- Overview of United States Senate Bill UD-750
- Presentation of United States Senate Bill UD-750
- Vote on United States Senate Bill UD-750
- Closing Remarks

Bureau of Safety

Bill SF-159

- Overview of United States Senate Bill SF-159
- Presentation of United States Senate Bill SF-159
- Vote on United States Senate Bill SF-159

Bill SF-753

- Overview of United States Senate Bill SF-753
- Presentation of United States Senate Bill SF-753
- Vote on United States Senate Bill SF-753
- Closing Remarks

Task 2B: Please modify the document that you just created so that it looks like the document below.

Agenda for United States Senate, 1 October

Bureau of Transportation

Bill TR-987

- *Overview of United States Senate Bill TR-987*
- *Presentation of United States Senate Bill TR-987*
- *Vote on United States Senate Bill TR-987*

Bill TR-236

- *Overview of United States Senate Bill TR-236*
- *Presentation of United States Senate Bill TR-236*
- *Vote on United States Senate Bill TR-236*
- *Closing Remarks*

Bureau of Housing

Bill HS-345

- *Overview of United States Senate Bill HS-345*
- *Presentation of United States Senate Bill HS-345*
- *Vote on United States Senate Bill HS-345*

Bill HS-632

- **Overview of United States Senate Bill HS-632**
- *Presentation of United States Senate Bill HS-632*
- *Vote on United States Senate Bill HS-632*
- *Closing Remarks*

Bureau of Urban Development

Bill UD-696

- *Overview of United States Senate Bill UD-696*
- *Presentation of United States Senate Bill UD-696*
- *Vote on United States Senate Bill UD-696*

Bill UD-750

- *Overview of United States Senate Bill UD-750*
- *Presentation of United States Senate Bill UD-750*
- *Vote on United States Senate Bill UD-750*
- *Closing Remarks*

Bureau of Safety

Bill SF-159

- *Overview of United States Senate Bill SF-159*
- *Presentation of United States Senate Bill SF-159*
- *Vote on United States Senate Bill SF-159*

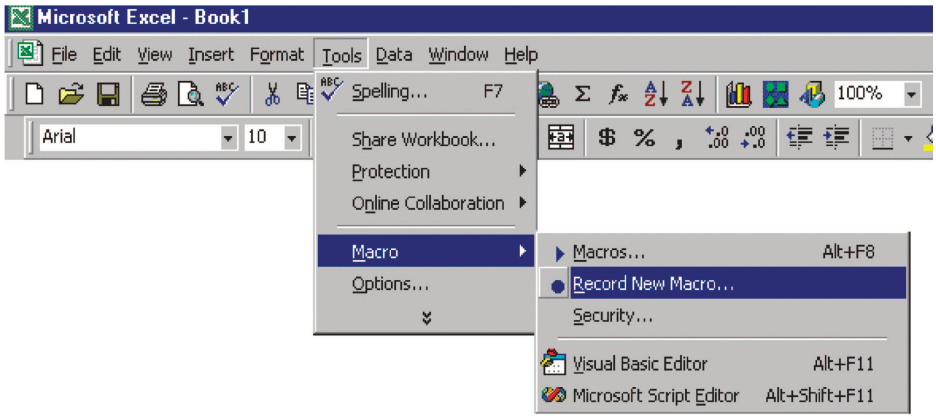
Bill SF-753

- **Overview of United States Senate Bill SF-753**
- *Presentation of United States Senate Bill SF-753*
- *Vote on United States Senate Bill SF-753*
- *Closing Remarks*

Task 3: MS Excel

Step 1: Please open the spreadsheet named “temperatures.xls” located on the desktop.

Step 2: Select Tools → Macro → Record New Macro, as shown below:



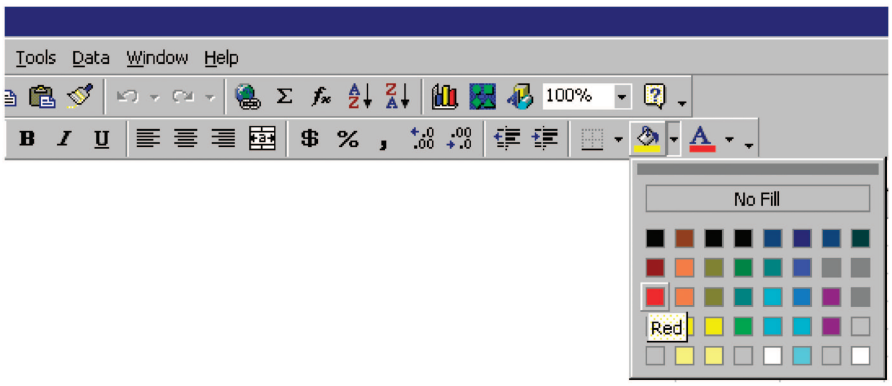
Step 3: Press OK in the Record Macro dialog box.

Please raise your hand if you have any problems.

The "temperatures.xls" spreadsheet contains the daily temperature at 12:00 noon in Seattle over many years for the month of August. Please perform the following tasks using the spreadsheet:

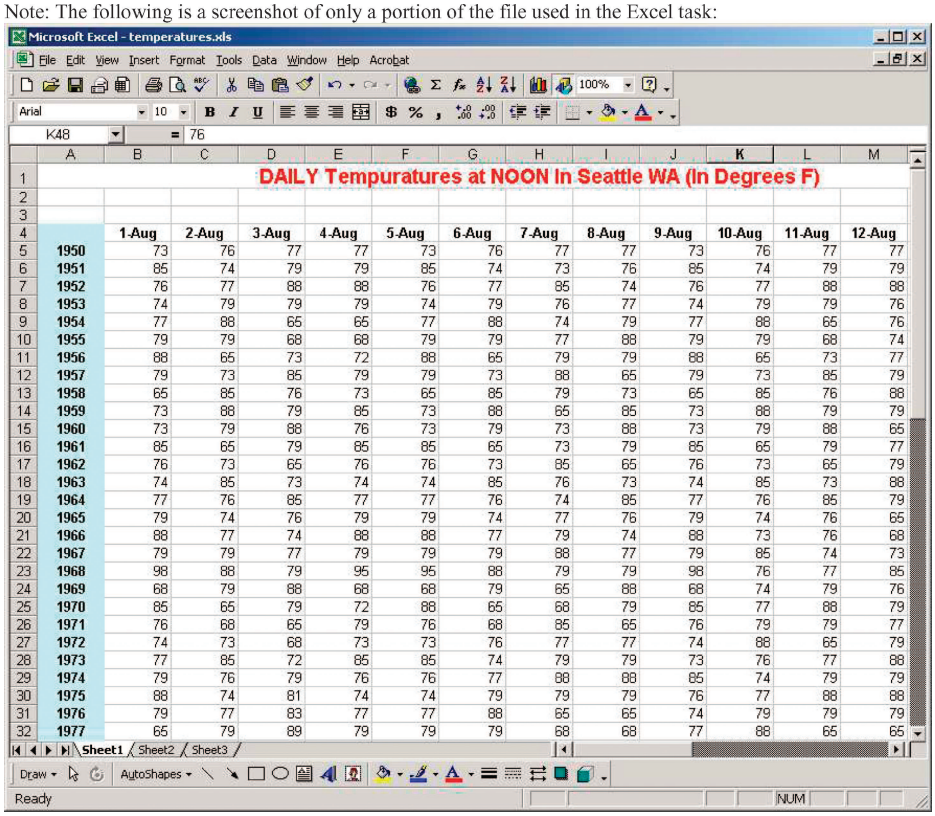
Task 3A. Which of the following pairs of days was hotter?
August 7, 1954 or August 7, 1995 (paint the cell with the hotter temperature red)
August 4, 1950 or August 4, 1985 (paint the cell with the hotter temperature red)
August 1, 1951 or August 6, 1951 (paint the cell with the hotter temperature red)
August 2, 1959 or August 5, 1995 (paint the cell with the hotter temperature red)

For the above tasks, use the **fill color** *icon on the toolbar to paint a cell as shown below:*



Task 3B. There was a technical problem in recording some of the temperatures. Three consecutive days in a year are missing. Please find them and paint them blue.
Task 3C. Create a set of cells containing the average temperature for the month of August for each of the years 1950-1999. Paint yellow the cell with the highest average temperature.
Task 3D. Scientists estimate that the temperatures for the three missing days were 98, 99, and 100 degrees. With this new information, what is the highest average temperature in the month of August? Please paint the cell brown.
Task 3E. Please create a visual representation of the daily temperatures for the month of August in the year 1950.

Note: The following is a screenshot of only a portion of the file used in the Excel task:

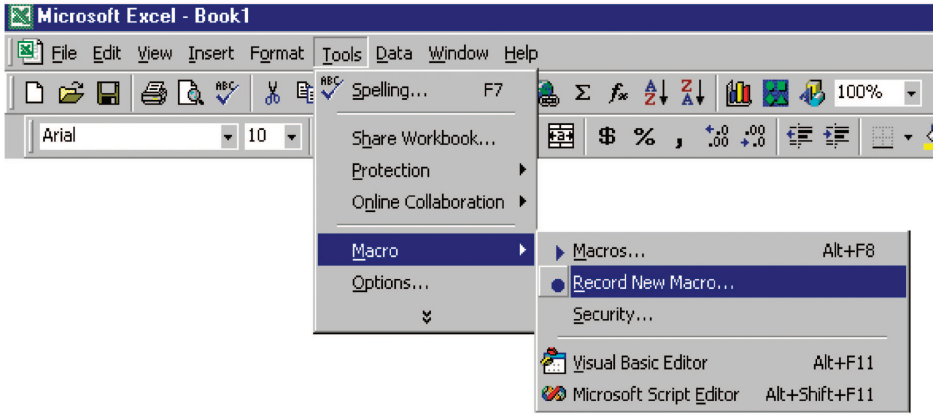


APPENDIX C. RETENTION TEST TASKS

Task 1 - Excel

Step 1: Please open the spreadsheet named "grades.xls" located on the desktop.

Step 2: Select Tools → Macro → Record New Macro, as shown below:



Step 3: Press OK in the Record Macro dialog box.

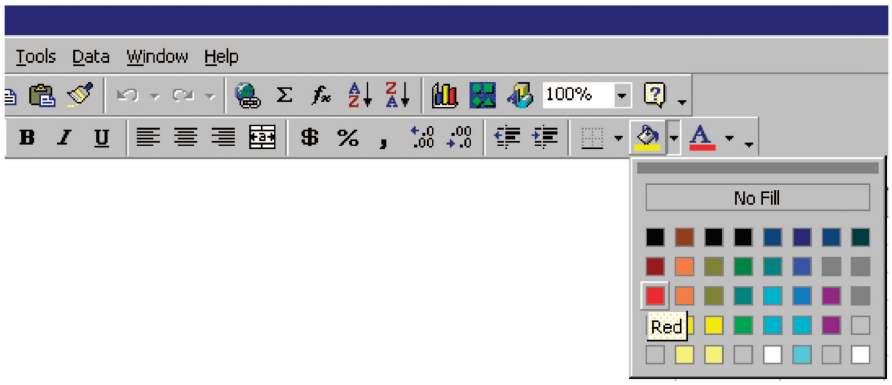
Please raise your hand if you have any problems.

The "grades.xls" spreadsheet contains the weekly quiz grades of students with IDs from A-1 to A-31. Please perform the following tasks using the spreadsheet:

Task 1A. In each of the following pairs of cells, which grade was higher?

- A-3's grade in week-1, or A-3's grade in week-45? *(paint the cell with the higher grade red)*
- A-7's grade in week-2, or A-7's grade in week-32? (paint the cell with the higher grade red)
- A-3's grade in week-2, or A-10's grade in week-34? (paint the cell with the higher grade red)
- A-9's grade in week-1, or A-12's grade in week-50? (paint the cell with the higher grade red)

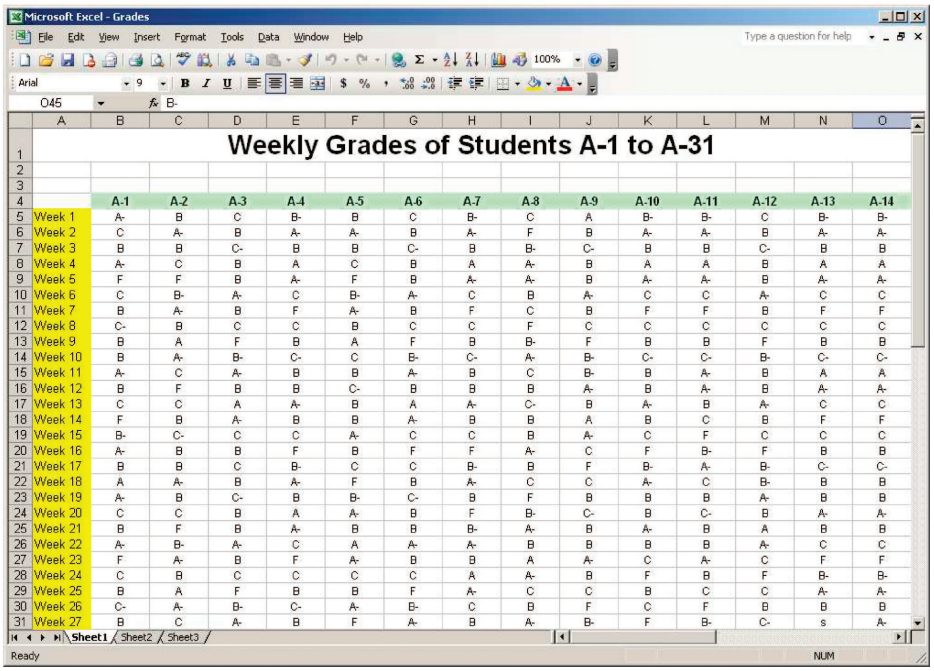
For the above tasks, use the **fill color** *icon on the toolbar to paint a cell as shown below:*



Go to **Sheet2** of the spreadsheet *(Click on the Sheet2 tab at the bottom of the sheet)*.

Task 1B. Sheet2 contains the numeric grades of another group of students B1 – B25. Create a set of cells containing the average grades for each of the students over 52 weeks. Paint yellow the cell with the highest average grade.
Task 1C. A student missed three consecutive quizzes marked by question marks. Please find them and paint them blue.
Task 1D. Add zeros for the missing grades. With this new information, which student had the highest average grade? Please paint brown the cell with the new highest average grade.
Task 1E. Please create a visual representation of the average grades for all the students.

Note: The following is a screenshot of only a portion of the file used in the Excel task:



		A-1	A-2	A-3	A-4	A-5	A-6	A-7	A-8	A-9	A-10	A-11	A-12	A-13	A-14
5	Week 1	A-	B	C	B-	B	C	B-	C	A	B-	B-	C	B-	B-
6	Week 2	C	A-	B	A-	A-	B	A-	F	B	A-	A-	B	A-	A-
7	Week 3	B	B	C-	B	B	C-	B	B-	C-	B	B	C-	B	B
8	Week 4	A-	C	B	A	C	B	A	A-	B	A	A	B	A	A
9	Week 5	F	F	B	A-	F	B	A-	A-	B	A-	A-	B	A-	A-
10	Week 6	C	B-	A-	C	B-	A-	C	B	A-	C	C	A-	C	C
11	Week 7	B	A-	B	F	A-	B	F	C	B	F	F	B	F	F
12	Week 8	C-	B	C	C	B	C	C	F	C	C	C	C	C	C
13	Week 9	B	A	F	B	A	F	B	B-	F	B	B	F	B	B
14	Week 10	B	A-	B-	C-	C	B-	C-	A-	B-	C-	C-	B-	C-	C-
15	Week 11	A-	C	A-	B	B	A-	B	C	B-	B	A-	B	A	A
16	Week 12	B	F	B	B	C-	B	B	B	A-	B	A-	B	A-	A-
17	Week 13	C	C	A	A-	B	A	A-	C-	B	A-	B	A-	C	C
18	Week 14	F	B	A-	B	B	A-	B	B	A	B	C	B	F	F
19	Week 15	B-	C-	C	C	A-	C	C	B	A-	C	F	C	C	C
20	Week 16	A-	B	B	F	B	F	F	A-	C	F	B-	F	B	B
21	Week 17	B	B	C	B-	C	C	B-	B	F	B-	A-	B-	C-	C-
22	Week 18	A	A-	B	A-	F	B	A-	C	C	A-	C	B-	B	B
23	Week 19	A-	B	C-	B	B-	C-	B	F	B	B	B	A-	B	B
24	Week 20	C	C	B	A	A-	B	F	B-	C-	B	C-	B	A-	A-
25	Week 21	B	F	B	A-	B	B	B-	A-	B	A-	B	A	B	B
26	Week 22	A-	B-	A-	C	A	A-	A-	B	B	B	B	A-	C	C
27	Week 23	F	A-	B	F	A-	B	B	A	A-	C	A-	C	F	F
28	Week 24	C	B	C	C	C	C	A	A-	B	F	B	F	B-	B-
29	Week 25	B	A	F	B	B	F	A-	C	C	B	C	C	A-	A-
30	Week 26	C-	A-	B-	C-	A-	B-	C	B	F	C	F	B	B	B
31	Week 27	B	C	A-	B	F	A-	B	A-	B-	F	B-	C-	s	A-

Task 2: MS Word

Task 2A: Please use MS Word to create the document below (Fonts need not be exactly the same as those shown). Design the document so that its appearance is easy to modify.

2001 Summer Courses

PS-435: Research and Technology in the Humanities

Teaching Staff

- Instructor: John Trimble
- Teaching Assistant: Gary Bell
- Guest Lecturer: Jim Smith

Equipment

- Projection device: IBM Laptop
- Microphone: Clip-on
- Pointing device: Laser pointer

CS-435: Data Structures

Teaching Staff

- Instructor: Michelle Faye
- Teaching Assistant: Sam Smith

Equipment

- Projection device: IBM Laptop
- Microphone: Clip-on
- Pointing device: Telescopic pointer

CS-296: Systems, Networks, and Webs

Teaching Staff

- Instructor: George Thomson
- Teaching Assistant: Sandy Lincoln, Peter Olson
- Guest Lecturer: Simon Sung

Equipment

- Projection device: IBM Laptop
- Microphone: Table-top
- Pointing device: Laser pointer

AC-296: History of Architecture

Teaching Staff

- Instructor: Tom Swift
- Teaching Assistant: Gary Bell
- Guest Lecturer: Jim Baker

Equipment

- Projection device: 35mm slide projector
- Microphone: Clip-on
- Pointing device: Laser pointer

Task 2B: Please modify the document that you just created so that it looks like the document below.

2001 Summer Courses

PS-435: Research and Technology in the Humanities

Teaching Staff

- *Instructor: John Trimble*
- *Teaching Assistant: Gary Bell*
- *Guest Lecturer: Jim Smith*

Equipment

- *Projection device: IBM Laptop*
- *Microphone: Clip-on*
- *Pointing device: Laser pointer*

CS-435: Data Structures

Teaching Staff

- *Instructor: Michelle Faye*
- *Teaching Assistant: Sam Smith*

Equipment

- *Projection device: IBM Laptop*
- *Microphone: Clip-on*
- *Pointing device: Telescopic pointer*

CS-296: Systems, Networks, and Webs

Teaching Staff

- *Instructor: George Thomson*
- *Teaching Assistant: Sandy Lincoln, Peter Olson*
- *Guest Lecturer: Simon Sung*

Equipment

- *Projection device: IBM Laptop*
- *Microphone: Table-top*
- *Pointing device: Laser pointer*

AC-296: History of Architecture

Teaching Staff

- *Instructor: Tom Swift*
- *Teaching Assistant: Gary Bell*
- *Guest Lecturer: Jim Baker*

Equipment

- *Projection device: 35mm slide projector*
- *Microphone: Clip-on*
- *Pointing device: Laser pointer*

APPENDIX D. SEGMENT OF THE TEACHING GUIDE FOR DREAMWEAVER

Simple Tasks: Tables
General Table Commands: Insert, Define Attributes, Resize

Motivating Example: Inserting a Table

- Open the document 'music-after.html' (located in the 'music' folder) in Explorer.
- This is another page of the Web site we saw last time to sell books, and music we discussed last time. This page shows music for sale.
- Can I modify this page? *(no – you have to use Dreamweaver® to modify a Webpage)*.
- Information on a Web page is best laid out using tables.
- This page is organized using tables. Tables allow you to align text and images easily. So if I add any information to this cell, it will wrap around and extend the table automatically.
- Tables in Dreamweaver® are similar to tables in Word® and in Excel®.
- Let's take a closer look at this table.
- Open the document 'table.htm' (located in the 'music' folder) in Dreamweaver®.
- How many rows and columns do you think are in this table?
- What is special about the alternate rows?
- To create this table, I have to use the Insert Table menu item. (Explain rows, columns, width and border)

Practice

- Open a new file on your computer. (This might need to be demonstrated. Click on File → New and select 'basic page'. Then click 'Create'.)
- Save the file under the name 'Tables'.
- Insert a new table with 7 rows, 2 columns, 100% width, 0 padding, 0 spacing, and 0 border.

  . . .

Complex Tasks

Problem-Solving Example: Make Organizations Known to the Computer

- Open the document 'godiva.html' in a browser.
- This document contains Valentine chocolates!
- Describe how you would design a Web page to display these chocolates.
- Efficient method:
  Organize content in tables. Open 'godiva.html' in Dreamweaver
  - Why? With tables, information is organized and easy to modify. For example, I can change the dimensions of this table or add content and the overall layout is still maintained.

- Demonstrate some modifications
- In general, as we have noted:
  Make organizations known to the computer.
  (Strategy 6 in handout)

APPENDIX E. RETENTION TEST TASKS

Table V. Pair-wise Chi-square comparisons for each strategy between the Command and Strategy groups in the CMU-1 and CMU-2 experiments. Grayed cells show statistically significant differences at the .05 level between the command and strategy group.

Strategy opportunities		CMU-1	CMU-2
UNIX			
A.	Reuse and modify groups of objects (*mv* *)	Chi-square (df=1, n=90) = 26.16, p<.001	Chi-square (df=1, n=36) = 4.12, p<.05
B.	Check original before making copies/operating on objects (*ls* *)	Chi-square (df=1, n=90) = 5.75, p<.05	Chi-square (df=1, n=36) = 0.92, p=.34
MSWord			
C.	Reuse and modify groups of objects (*copy/paste*)	Chi-square (df=1, n=90) = 4.52, p<.05	Chi-square (df=1, n=36) = 6.29, p<.05
D.	Make organizations known to the computer (*lists*)	Chi-square (df=1, n=90) = 0.88, p=.35	Chi-square can not be calculated because 100% of subjects used the strategy
E.	Make dependencies known to the computer (*styles*)	Chi-square (df=1, n=90) = 31.68, p<.001	Chi-square (df=1, n=36) = 10.98, p<.001
F.	Exploit dependencies to generate variations (*modify styles*)	Chi-square (df=1, n=90) = 25.48, p<.001	Chi-square (df=1, n=36) = 5.40, p<.05
MSExcel			
G.	View parts of spread out information simultaneously on the screen (*split window*)	Chi-square (df=1, n=90) = 23.29, p<.001	Chi-square (df=1, n=36) = 13.22, p<.001
H.	View relevant information, do not view irrelevant information (zoom)	Chi-square (df=1, n=90) = 5.40, p<.05	Chi-square (df=1, n=36) = 0.31, p=.58
I.	Make dependencies known to the computer (*formulas*)	Chi-square can not be calculated because 100% of subjects used the strategy	Chi-square (df=1, n=36) = 1.21, p=.27
J.	Reuse and modify groups of objects (*drag across cells*)	Chi-square can not be calculated because 100% of subjects used the strategy	Chi-square can not be calculated because 100% of subjects used the strategy
K.	Exploit dependencies to generate variations (*modify dependent cells*)	Chi-square (df=1, n=90) = 1.722, p=.19	Chi-square (df=1, n=36) = 2.52, p=.11
L.	Generate new representations from existing representations (*charts*)	Chi-square (df=1, n=90) = 0.50, p=.58	Chi-square (df=1, n=36) = 5.54, p<.05

Table VI. Pair-wise Chi-square comparisons for each strategy between the command + Ap-specific and Strategy groups in the UM-1 experiment. Grayed cells show statistically significant differences at the .05 level. Strategy opportunity L was designed as an extra-credit question.

Strategy Opportunities		UM-1 (Post-test)
MSWord		
D.	Make organizations known to the computer (*lists*)	Chi-square can not be calculated because 100% of subjects used the strategy
E.	Make dependencies known to the computer (*styles*)	Chi-square (df=1, n=50) = 0.35, p=.55
F.	Exploit dependencies to generate variations (*modify styles*)	Chi-square (df=1, n=50) = 0.32, p=.57
MSExcel		
G.	View parts of spread out information simultaneously on the screen (*split window*)	Chi-square (df=1, n=50) = 3.99, p<.05
H.	View relevant information, do not view irrelevant information (*zoom*)	Chi-square (df=1, n=50) = 0.16, p=.68
I.	Make dependencies known to the computer (*formulae*)	Chi-square (df=1, n=50) = 1.12, p=.29
J.	Reuse and modify groups of objects (*Drag across cells*)	Chi-square (df=1, n=50) = 1.08, p=.30
K.	Exploit dependencies to generate variations (*modify dependent cells*)	Chi-square (df=1, n=50) = 0.94, p=.33
L.	Generate new representations from existing representations (*charts*)	Chi-square (df=1, n=50) = 0.33, p=.56

ACKNOWLEDGMENTS

We thank B. John, who guided the early stages of this research at Carnegie Mellon University, and G. Olson and J. King for encouraging the deployment and testing of strategy-based instruction at the University of Michigan. Furthermore, we thank M. Basket, N. Bos, S. Carpenter, T. Finholt, U. Flemming, O. Frost, J. Lee, J. Olson, C. Quintana, B. Rogers, A. Sue, L. Strodtman, R. Thomas, G. Vallabha, L. Weber, and the anonymous reviewers for their contributions.

REFERENCES

ANDERSON, J. R. 2000. *Learning and Memory: An Integrated Approach*. 2nd Ed. John Wiley & Sons, New York, NY. (Chap. 7, 232–272).

ANDERSON, J. R. AND MILSON, R. 1989. Human memory: An adaptive perspective. *Psych. Rev. 96*, 703–719.

ANDERSON, J. R., CORBETT, A. T., KOEDINGER, K. R., AND PELLETIER, R. 1995. Cognitive tutors: Lessons learned. *J. Learn. Sci. 4*, 2, 167–207.

BEREITER, C. 2002. *Education and the Mind in the Knowledge Age*. Erlbaum, Mahwah, NJ.

BHAVNANI, S. K. 1998. How Architects draw with computers: A cognitive analysis of real-world CAD interactions, Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA.

BHAVNANI, S. K. 2001. Important cognitive components of domain-specific search knowledge. In *Proceedings of the Text Retrieval Conference* (TREC'01). 571–578.

BHAVNANI, S. K. 2005. Why is it difficult to find comprehensive information? Implications of information scatter for search and design. *J. Amer. Soc. Inform. Sci. Tech. 56*, 9, 989–1003.

BHAVNANI, S. K., BICHAKJIAN, C. K., JOHNSON, T. M., LITTLE, R. J., PECK, F. A., SCHWARTZ, J. L., AND STRECHER, V. J. 2006. Strategy hubs: Domain portals to help find comprehensive information. *J. Amer. Soc. Inform. Sci. Tech. 57*, 1, 4–24.

BHAVNANI, S. K. AND JOHN, B. E. 1996. Exploring the unrealized potential of computer-aided drafting. In *Proceedings of Computer-Human Interaction (CHI'96)*. 332–339.

BHAVNANI, S. K. AND JOHN, B. E. 1998. Delegation and circumvention: Two faces of efficiency. In *Proceedings of Computer-Human Interaction (CHI'98)*. 273–280.

BHAVNANI, S. K. AND JOHN, B. E. 2000. The strategic use of complex computer systems. *Hum.-Comput. Interac. 15*, 107–137.

BHAVNANI, S. K., FLEMMING, U., FORSYTHE, D., GARRETT, J. H., SHAW, D. S., AND TSAI, A. 1996. CAD usage in an architectural office: From observations to active assistance. *Autom. Construc. 5*, 243–255.

BHAVNANI, S. K., JOHN, B. E., AND FLEMMING, U. 1999. The strategic use of CAD: An empirically inspired, theory-based course. In *Proceedings of Computer-Human Interaction (CHI'99)*. 42–49.

BHAVNANI, S. K., REIF, F., AND JOHN, B. E. 2001. Beyond command knowledge: Identifying and teaching strategic knowledge for using complex computer applications. In *Proceedings of Computer-Human Interaction (CHI'01)*. 229–236.

BORKO, H. AND LIVINGSTON, C. 1989. Cognition and improvisation: Differences in mathematics instruction by expert and novice teachers. *Amer. Educ. Resear. J. 26*, 4, 473–498.

BOSSOCK, M. AND HOLYOAK, K. J. 1989. Interdomain transfer between isomorphic topics in algebra and physics. *J. Exper. Psych.: Learn., Memory, Cognition 15*, 153–166.

BROWN, A. 1992. Design experiments: theoretical and methodological challenges in creating complex interventions in classroom settings. *J. Learn. Sci. 2*, 141–178.

BROWN, A. AND PALINSCAR, A. 1989. Guided, cooperative learning and individual knowledge acquisition. In L. Resnick Ed., *Knowing, Learning and Instruction*, Lawrence Erlbaum, Hillsdale, MA.

BROWN, M. AND EDELSON, D. C. 2001. Teaching by design: Curriculum design as a lens on instructional practice. *Annual Meeting of the American Educational Research Association*. Seattle, WA.

CARD, S. K., MORAN, T. P., AND NEWELL, A. 1983. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, Hillsdale, MA.

CARROLL, J. M. AND AARONSON, A. P. 1988. Learning by doing with simulated intelligent help. *Comm. ACM 31*, 9, 1064–1079.

CARROLL, J. M. AND ROSSON, M. B. 1987. The paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, J. M. Carroll, Ed., MIT Press, Cambridge, MA. 80–111.

CARROLL, J. M., SMITH-KERKER, P. L., FORD, J. R., AND MAZUR-RIMETZ, S. A. 1987. The minimal manual. *Hum.-Comput. Interac. 3*, 123–153.

CHARMAN, S. C. AND HOWES, A. 2003. The adaptive user: An investigation into the cognitive and task constraints on the generation of new methods. *J. Exper. Psych.: Appl. 9*, 236–248.

CHARNEY, D., REDER, L., AND KUSBIT, G. 1990. Goal setting and procedure selection in acquiring computer skills: A comparison of tutorials, problem solving, and learner exploration. Cognition Instruct. 7, 4, 323–342.

COLLINS, A., BROWN, J. S., AND NEWMAN, S. E. 1989. Cognitive apprenticeship: teaching the crafts of reading, writing, and mathematics. In *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, L. B. Resnick Ed., Erlbaum, Hillsdale, NJ. 453–494.

CRAGG, P. B. AND KING, M. 1993. Spreadsheet modeling abuse: An opportunity for OR? *J. Operat. Resear. Soc. 44*, 743–752.

EISENBERG, M. B. AND JOHNSON, D. 2002. Learning and teaching information technology: Computer skills in context. *ERIC Digest*. www.big6.com/showarticle.php?id=82.

ENGELMANN, S. 1980. *Direct Instruction*. Educational Technology Publications, Englewood Cliffs, NJ.

FLEMMING, U., BHAVNANI, S. K., AND JOHN, B. E. 1997. Mismatched metaphor: User vs. system model in computer-aided drafting. *Design Studies 18*, 349–368.

FONG, G. T., KRANTZ, D. H., AND NISBETT, R. E. 1986. The effects of statistical training on thinking about everyday problems. *Cognitive Psych. 18*, 253–292.

FU, W. AND GRAY, W. D. 2004. Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. *Cognitive Sci. 28*, 6, 901–935.

GICK, M. L. AND HOLYOAK, K. J. 1983. Schema induction and analogical transfer. *Cognitive Psych. 15*, 1–38.

GOLDWEBER, M., BARR, J., AND LESKA, C. 1994. A new perspective on teaching computer literacy. In *Proceedings of the Technical Symposium on Computer Science Education (SIGSCE'94)*. 131–135.

GONG, R. AND ELKERTON, J. 1990. Designing minimal documentation using a GOMS model: a usability evaluation of an engineering approach. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'90)*. 99–106.

HALARIS, A. AND SLOAN, L. 1985. Towards a definition of computer literacy for the liberal arts environment. In *Proceedings of the Technical Symposium on Computer Science Education (SIGSCE'85)*. 320–326.

HARTER, S. P. AND PETERS, A. R. 1985. Heuristics for online information retrieval: a typology and preliminary listing. *Online Rev. 9*, 5, 407–424.

HINTZMAN, D. L. 1969. Recognition time: Effects of recency, frequency and the spacing of repetitions. *J. Exper. Psych. 79*, 192–194.

HOFFMAN, M. AND BLAKE, J. 2003. Computer literacy: Today and tomorrow. *J. Comput. Small Colleges 18*, 5, 221–233.

INTERNATIONAL SOCIETY FOR TECHNOLOGY IN EDUCATION (ISTE). 1999. National Educational Technology Standards for Students. http://cnets.iste.org.

JOHN, B. AND KIERAS, D. 1996. The GOMS family of user interface analysis techniques: comparison and contrast. *Trans. Comput.-Hum. Interac. 3*, 4, 320–351.

KLAHR, D. AND CARVER, S. M. 1988. Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psych. 20*, 352–404.

LEE, W. O. AND BARNARD, P. J. 1993. Precipitating change in system usage by function revelation and problem reformulation. In *Proceedings of Human Computer Interaction (HCI'93)*. 35–47.

LUCHINS, A. S. AND LICHINS, E. H. *Wertheimer's Seminars Revisited: Problem Solving and Thinking*. State University of New York: Albany, NY.

MARSH, C. 2007. Strategic knowledge of computer applications: The key to efficient use. *J. Issues Inform. Sci. Inform. Techn. 4*, 268–276.

MCCADE, J. 2001. Technology education and computer literacy. *Techn. Teach.* 9–13.

MYERS, J. P. 1989. The new generation of computer literacy. In *Proceedings of the Technical Symposium on Computer Science Education (SIGSCE'89)*. 177–181

NATIONAL RESEARCH COUNCIL 2000. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. National Academy Press.

NICHOLLS, J. G. 1989. *The Competitive Ethos and Democratic Education*. Harvard University Press, Cambridge, MA.

NILSEN, E., JONG. H., OLSON, J., BIOLSI, I., AND MUTTER, S. 1993. The growth of software skill: A longitudinal look at learning and performance. In *Proceedings of Computer-Human Interaction (CHI'93)*. 149–156.

NOLEN, S. B. 1996. Why study? How reasons for learning influence strategy selection. *Educ. Psych. Rev. 8*, 4, 335–355.

NOLEN, S. B. 2003. Learning environment, achievement, and motivation in high school science. *J. Resear. Sci. Teach. 40*, 347–168.

OLSON, J. R. AND NILSEN, E. 1988. Analysis of the cognition involved in spreadsheet software interaction. *Hum. Comput. Interac. 3*, 309–349.

PALINSCAR, A. 1998. Social constructivist perspectives on teaching and learning. In *Annual Review of Psychology 49*, J. T. Spence, J. M. Darley, and D. J. Foss, D. J., Eds. Annual Reviews, Palo Alto, CA. 345–375.

PALINSCAR, A. AND BROWN, A. 1984. Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition Instruct. 1*, 117–175.

PAYNE, S. J., HOWES, A., AND READER, W. R. 2001. Adaptively distributing cognition: a decision-making perspective on human-computer interaction. *Behav. Inform. Techn. 20*, 5, 339–346.

PINTRICH, P. R. AND SCHUNK, D. H. 1996. *Motivation in Education: Theory, Research, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

ROSSON, M. 1983. Patterns of experience in text editing. In *Proceedings of Computer-Human Interaction (CHI'83)*. 171–175.

SCHOENFELD, A. H. 1985. *Mathematical Problem Solving*. Academic Press, New York, NY.

SELLARS, H. L. 1988. Why a course in computer literacy? *SIGCSE Bull. 20*, 2, 58–64.

SHAW, S. AND POLOVINA, S. 1999. Practical experiences of, and lessons learnt from, Internet technologies in higher education. *Educ. Techn. Soc. 2*, 3, 16–23.

SINGLEY, M. K. AND ANDERSON, J. R. 1989. *The Transfer of Cognitive Skill*. Harvard University Press, Cambridge, MA.

THOMAS, R. AND FOSTER, M. 2001. A pilot study of teaching the strategic use of common computer applications. In *Proceedings of Australian User Interface Conference*. 85–92.

UNDERWOOD, B. J. 1969. Some correlates of item repetition in free-recall learning. *J. Verbal Learn. Verbal Behav. 8*, 83–94.

YINGER, R. J. 1987. By the seat of your pants: An inquiry into improvisation and teaching. *Annual Meeting of the American Educational Research Association*, Washington DC.

Received May 2005; revised September 2006; accepted August 2007 by Marti Hearst