COGNITION AND INSTRUCTION, 1995, *13*(2), 221–252
Copyright © 1995, Lawrence Erlbaum Associates, Inc.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Training in Self-Explanation and Self-Regulation Strategies: Investigating the Effects of Knowledge Acquisition Activities on Problem Solving

Katerine Bielaczyc, Peter L. Pirolli, and Ann L. Brown
School of Education
University of California, Berkeley

Previous research has found positive correlations between particular strategies students use while studying to explain instructional materials to themselves and student performance on associated problem-solving tasks (Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). In the study reported here, we investigate the causal nature of this relation. This was accomplished by identifying a set of self-explanation and self-regulation strategies used by high-performance students in our earlier studies. We used strategy training to manipulate students' application of these strategies and examined the impact of their use on student explanations and performance. Twenty-four university students with no prior programming experience worked through a sequence of programming lessons. Following introductory lessons, participants received interventions involving explicit training in the strategies (instructional group) or received a similar set of interventions but no explicit training (control group). The instructional group showed significantly greater gains than the control group in the use of self-explanation and self-regulation strategies from the pre- to postinterventions lessons. Increased strategy application was accompanied by significantly greater performance gains. The results indicate that the particular self-explanation and self-regulation strategies used in training contribute to learning and problem-solving performance.

Students may be able to follow each line of a geometry proof as a teacher writes it on the board, be able to understand principles underlying the steps of a physics demonstration, or comprehend the main points of a biology text and yet not be

Requests for reprints should be sent to Peter Pirolli, Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304.

able to solve related problems when asked to do so on their own. There is a difference between comprehending and learning from instructional materials and presentations (vanDijk & Kintsch, 1983). An important aspect of learning is how the concepts and principles of a domain are initially encoded and how these encodings impact solving problems based on that knowledge. We propose that there are substantive and interesting individual differences in the kinds of learning and metacognitive strategies used by learners to acquire knowledge in complex problem-solving domains. We focus our study on self-explanation and self-regulation strategies; how learners explain instructional materials to themselves and how they monitor their study activities and states of comprehension. Our particular interest is in the relation of these strategies to cognitive skill acquisition and problem-solving performance.

Recent studies of physics (Chi, Bassok, Lewis, Reimann, & Glaser, 1989) and programming (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994) suggest that the acquisition of cognitive skill is affected not only by the quantity but by the quality of self-explanations produced by learners. These studies found that particular characteristics of the self-explanations made by students while studying instructional materials correlated with the students' subsequent problem-solving performance. The high-performance students were found to use certain self-explanation and self-regulation strategies in constructing their explanations. For instance, when high performers studied the examples in the instruction, they typically connected example features to concepts that had been introduced in the text.

An important question concerning the results of Chi et al. (1989), Pirolli and Bielaczyc (1989), and Pirolli and Recker (1994) is whether the self-explanation and self-regulation strategies differentiating high- and low-performance students play a causal role in problem-solving performance. The relations between the observed strategies and problem-solving performance have been identified through correlational but not through experimental analyses. The earlier studies did not perform independent manipulations of self-explanations in order to examine their effects on subsequent performance. In the present study, explanation activities are manipulated by training students on a group of strategies related to higher programming performance. If the strategies do play a contributing role in the acquisition of cognitive skills, we would expect an improved use of the strategies to be accompanied by an improvement in problem-solving performance.

GENERAL FRAMEWORK FOR THE RESEARCH

The current work is part of an ongoing research effort to construct formal models of learning in rich problem-solving domains such as computer programming. The underlying framework for the research is aimed at integrating models of active, goal-oriented learning processes with theories of problem-solving and cognitive skill development (e.g., Anderson, 1987; Bereiter & Scardamalia, 1989;

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Brown, Bransford, Ferrara, & Campione, 1983). The educational context involves learning from expository texts and examples, followed by solving associated problems. We assume that learners use study strategies and prior knowledge to actively interpret and elaborate the instructional texts and examples. Such interpretive processes yield declarative knowledge about the presented concepts, procedures, and examples. Exercise problems typically involve a mix of familiar and novel tasks. In solving problems, learners use as many domain-specific skills as possible. At problem-solving impasses, learners resort to general (weak-method) problem-solving strategies. Problem solving is guided by declarative knowledge acquired from the text and examples.

In general, research in connection with this framework has concentrated on the latter part of the process: conversion of declarative knowledge into problem-solving skills and models of knowledge compilation (e.g., Anderson, 1987; Newell & Rosenbloom, 1981), the use of analogy to examples in problem solving (e.g., Pirolli, 1987; Pirolli & Anderson, 1985; Ross, 1980), and models of performance (e.g., Anderson, 1983; Newell & Simon, 1972). In recent years, the focus has broadened from investigating the development and use of problem-solving skills to include the earlier phase of learning when the concepts and principles of a domain are initially encoded (e.g., Chi et al., 1989; Eylon & Helfman, in press; Ferguson-Hessler & de Jong, 1990; Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994; Reder, Charney, & Morgan, 1986). Differences in student study strategies, instructional content, methods of presentation, and prior knowledge during this initial encoding phase have been found to be related to differences in problem solving. Individual differences in self-explanation and self-regulation strategies during the encoding phase are thought to affect the quality and effectiveness of the declarative knowledge that is the basis for the acquisition of domain-specific cognitive skills. Effective strategies should produce greater yields of higher quality declarative knowledge that can be used during problem solving.

STRATEGY TRAINING

The strategy-training interventions were designed to communicate specific self-explanation and self-regulation strategies to students. The main pedagogical features of the interventions were modeling and scaffolding techniques (Collins, Brown, & Newman, 1989). These methods allowed students to observe and to apply the self-explanation and self-regulation strategies in relevant learning situations. The specific strategies chosen for strategy training were based on the results of our previous research (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). The self-explanation strategies that had been found to relate to high programming performance fall into the following top-level categories: (a) texts: identify and elaborate the relations between the main ideas, (b) examples: determine both the form and meaning of the Lisp code, and (c) texts and examples: connect the concepts in the texts and the examples.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

These top-level categories are consistent with domain-general strategies for learning from instructional texts and examples. Categories A and C are typical of most reading comprehension programs (Baker & Brown, 1984a, 1984b; Brown, 1980; Brown, Armbruster, & Baker, 1985; Chipman, Segal, & Glaser, 1985). Category B may be thought of in domain-general terms as "identify the main features or points of an example and their underlying rationale or purpose." During training, the top-level strategy types were described in general terms. But in addition, the strategies were specifically situated in the context of learning Lisp programming by providing examples of how each strategy applied to learning from instructional materials on familiar Lisp topics. These specific examples were based on instances of strategy application by high-performance participants from our prior investigations (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). During training it was also emphasized that the strategies should be applied toward the domain-specific learning goal; being able to code one's own Lisp functions and to solve problems associated with a given lesson.

We assumed that students should be fully informed participants in the strategy-training program and that the strategy instructions should include metacognitive aspects of learning (Brown et al., 1983; Brown, Campione, & Day, 1981). Campione and Brown (1990) stressed that the metacognitive component of training is important in that it allows students to understand and take control of their own learning process. In our training interventions, students were not only instructed in the use of particular self-explanation strategies and the significance of strategic activity, but they were also instructed in the types of self-regulation strategies found to be used by high performers in our previous studies. This included strategies in the following top-level categories: (a) monitoring comprehension and learning activities and (b) clarifying and addressing comprehension failures.

The strategy training was performed within the context of a mini-course on Lisp programming. We chose this context in order to examine learning and strategy usage over time. It was also important to provide a challenging learning context that would motivate students to use self-explanation and self-regulation strategies and to make strategy training relevant to students' learning goals. Because our intent was to investigate the causal nature of a particular set of self-explanation and self-regulation strategies, ideally our two experimental groups should have differed only in the application of these specific strategies. To keep the investigative focus on the impact of the self-explanation and self-regulation strategies and to reduce the impact of other variables, we designed the experiment using the following design criteria:

1. Equate incoming domain knowledge of all participants: We did not want prior programming knowledge to be a source of learner differences. We required that participants have no prior programming experience. Because Lisp programming is a cognitively complex domain, however, participants were required to have previously taken advanced problem-solving courses in mathematics.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

2. Equate learning experiences and knowledge base in the domain for all participants: We wanted to provide participants with common knowledge in the basics of Lisp prior to the main experimental sessions. All participants completed a set of introductory lessons on Lisp programming. Participants received the same lesson materials and problem sets for all of the programming lessons.

3. Take account of each participant's own strategic activities and performance level: We wanted to take account of each participant's own strategic learning activities and performance level prior to any training interventions in order to examine the impact of the interventions relative to existing strategies and performance abilities. We collected data on participants' explanations while studying instructional materials for the lessons both before and after strategy training, along with participants' associated programming performance for these lessons. By collecting pretraining measures, we also had a basis for equating the initial explanation and performance levels of the two experimental groups.

4. Equate components of the training interventions received by both experimental groups. Participants were divided into two groups for experimental interventions: One group received interventions involving explicit training in the strategies related to high performance (instructional group) and the other received a similar set of interventions but no explicit training (control group). Akin to blind training studies found in the strategy-training literature (refer to Campione & Armbruster, 1984, for a review of different types of training studies), in our study the control group was exposed to the same learning models and lesson materials and performed the same explanation activities as the instructional group. In essence, the instructional group was an explicit training group, and the control was an implicit training group. We wanted the interventions received by the two groups to differ only in the explicitness in training of the specific self-explanation and self-regulation strategies under investigation. This was important in order to attribute between-group differences from before to after strategy training to the application of the trained strategies.

METHOD

Participants

There were 24 participants in the study. The instructional group had 11 participants; the control group had 13 participants. The participants were either university students or recent graduates. No participants had prior programming experience; all had completed at least one semester of college-level calculus. The participants were recruited through campus advertisements and were paid $5.00 per hour. Five participants did not continue after the introductory session of the study: Two participants had difficulty in giving verbal protocols during the pretest, and three participants made no programming errors on Lesson 1. Of the participants completing the study, one was dropped due to a mistake in the procedure.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Procedure and Materials

The experiment involved several phases (illustrated in Figure 1). The main focus of the study was on the final three phases: the pre- and postinterventions programming lessons and the instructional interventions sessions. The pre- and postinterventions lessons had a parallel form consisting of the following two stages: (a) the encoding stage, in which the participants studied and explained

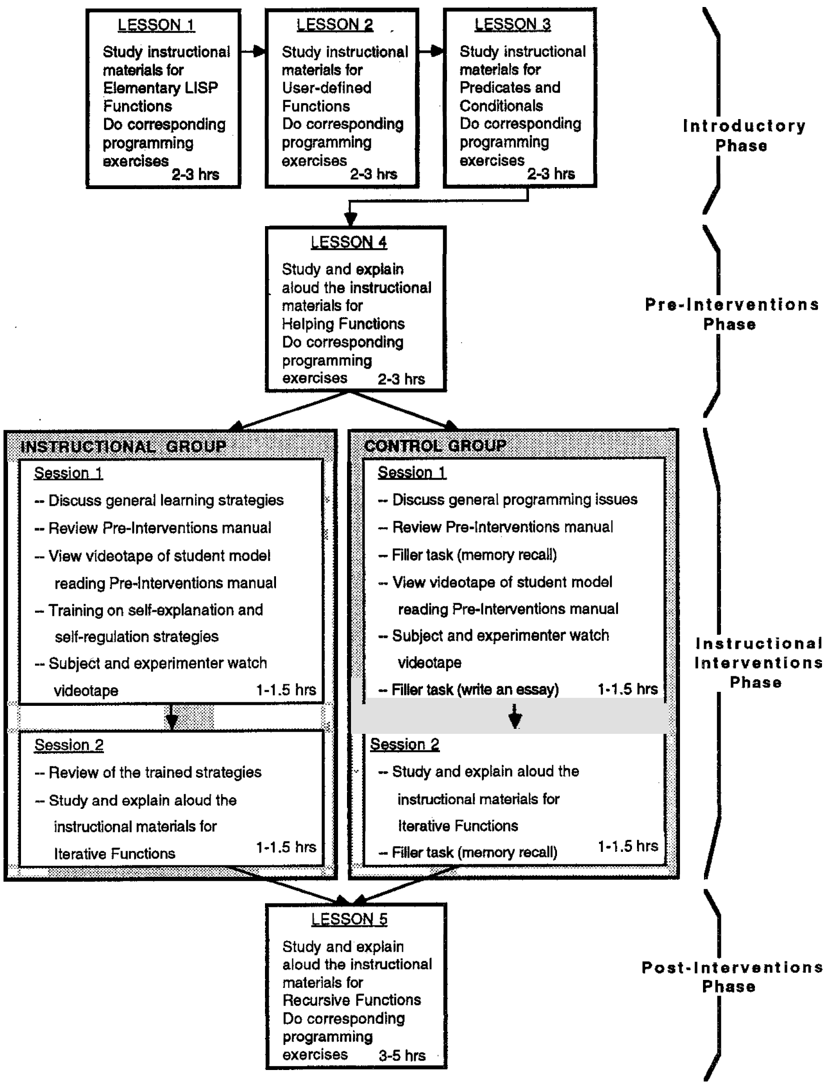
LESSON 1
Study instructional materials for Elementary LISP Functions
Do corresponding programming exercises 2-3 hrs
LESSON 2
Study instructional materials for User-defined Functions
Do corresponding programming exercises 2-3 hrs
LESSON 3
Study instructional materials for Predicates and Conditionals
Do corresponding programming exercises 2-3 hrs
Introductory Phase
LESSON 4
Study and explain aloud the instructional materials for Helping Functions
Do corresponding programming exercises 2-3 hrs
Pre-Interventions Phase
INSTRUCTIONAL GROUP
Session 1
-- Discuss general learning strategies
-- Review Pre-Interventions manual
-- View videotape of student model reading Pre-Interventions manual
-- Training on self-explanation and self-regulation strategies
-- Subject and experimenter watch videotape 1-1.5 hrs
Session 2
-- Review of the trained strategies
-- Study and explain aloud the instructional materials for Iterative Functions 1-1.5 hrs
CONTROL GROUP
Session 1
-- Discuss general programming issues
-- Review Pre-Interventions manual
-- Filler task (memory recall)
-- View videotape of student model reading Pre-Interventions manual
-- Subject and experimenter watch videotape
-- Filler task (write an essay) 1-1.5 hrs
Session 2
-- Study and explain aloud the instructional materials for Iterative Functions
-- Filler task (memory recall) 1-1.5 hrs
Instructional Interventions Phase
LESSON 5
Study and explain aloud the instructional materials for Recursive Functions
Do corresponding programming exercises 3-5 hrs
Post-Interventions Phase

FIGURE 1 The phases of the experiment.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

to themselves the instructional materials for the lesson and (b) the problem-solving stage, in which the participants solved the associated programming problems for the lesson. Our main interest was in the changes taking place between student explanation activities in the encoding stages and between performance in the problem-solving stages before and after strategy training.

The instructional context for the experiment involved participants working individually through a sequence of programming instruction using the Carnegie Mellon University Lisp Tutor (Reiser, Anderson, & Farrell, 1985). Although the use of an intelligent tutoring system is not common, the general structure of each lesson is typical of programming instruction: studying instructional materials and then solving associated programming exercises. The instructional manuals that the participants received for all lessons, in all phases of the study, were identical. The corresponding sets of programming exercises with the Lisp Tutor were also the same for each participant. Each exercise problem involved writing Lisp code to accomplish a specified task. The Lisp Tutor requires a participant to complete correctly each exercise problem before being able to continue to the next problem. The computerized prompts that the Lisp Tutor presented to each participant during the problem-solving stage had some variation, because the Tutor individualizes its interactions during a given problem solution, based on progress toward correctly completed code. One benefit of using the Lisp Tutor is that it allowed us to collect detailed information for each participant during the problem-solving stage. In addition, we were able to ensure that all participants reached a criterion level of correctly completing each problem in the exercise set for a given lesson.

The manuals for each lesson introduced new language features or programming constructs. Our development of the lesson manuals was guided by Kieras's (1985) model of technical manual design. Each lesson manual also included a review sheet summarizing previously introduced Lisp functions. The introductory manuals covered several basic issues of programming. The subsequent instructional manuals each focused on a specific type of function: helping functions, iterative functions, and recursive functions. These manuals were constructed to have parallel form consisting of the following components:

Page 1. A description of the structure of the type of function being introduced.
Page 2. An example function written in Lisp.
Page 3. A description of how the function type operates.
Page 4. An example evaluation trace of the code from page 2 as sample input values are processed.
Page 5. A description of design heuristics associated with writing this type of function.
Page 6. An example of the design heuristics using the code given on page 2.

The first, third, and fifth pages were textual descriptions presenting the central concepts of the lesson. The examples on the second, fourth, and sixth pages

Downloaded by [RMIT University Library] at 13:06 25 December 2017

contained associated instantiations of these concepts. The examples were unelaborated, and no direct references were given between text concepts and their instantiations in the examples. The first two pages of the postinterventions lesson manual are presented in the Appendix.

The seven sessions of the study were scheduled approximately every 2 days. To control and collect data on relevant learning experiences, participants were instructed not to study Lisp or other programming issues when they were not in the sessions. The participants also were not permitted to keep any of the instructional materials between sessions. During each programming lesson, participants were allowed to refer to the instructional materials of all previous lessons.

All sessions after the introductory phase were videotaped. The basic method used in gathering protocols from the participants was the same in all cases. Each session began with the experimenter reading to the participant an overview of the activities for the session and asking the participant to read aloud and to verbalize his or her thoughts while studying and working throughout the entire session. Experimenter intervention during the course of the sessions was minimal, typically restricted to prompting the participant to think aloud during prolonged periods of silence.

Phase 1: Introductory Phase

The purpose of the introductory phase was to provide the participants with common preliminary knowledge of Lisp programming. We wanted to control for incoming programming knowledge to the experimental sessions (Phases 2–4). The introductory phase also allowed participants to become familiar with the Lisp Tutor and the structure of the lessons prior to the experimental sessions.

In the introductory phase, participants proceeded through a set of initial experimental activities and a series of three introductory lessons on Lisp programming. The three introductory lessons covered elementary Lisp functions, user-defined functions, and predicates and conditionals, respectively. These lessons correspond to introductory topics typically found in Lisp programming textbooks (e.g., Anderson, Corbett, & Reiser, 1987; Winston & Horn, 1981). For each lesson, participants studied a lesson manual and solved a corresponding set of exercise problems using the Lisp Tutor.

The experiment was dependent on participants being both able and comfortable to think aloud while reading and problem solving. In the initial introductory session, a verbal protocol activity was given in order to screen out participants who reported discomfort in thinking aloud or who were inaudible and did not respond to requests to think aloud. Participants were videotaped and requested to think aloud while working through a short nonprogramming task. The task involved solving algebraic functions and logical reasoning problems. Prior to videotaping, participants were given a set of warm-up exercises meant to provide practice in giving verbal protocols (Ericsson & Simon, 1984). The warm-up

Downloaded by [RMIT University Library] at 13:06 25 December 2017

exercises consisted of simple arithmetic problems. The experimenter first modeled the verbal protocol process by thinking aloud while working through a sample warm-up problem.

Phase 2: Preinterventions Phase

The preinterventions phase was used to collect data on the explanations and programming performance of all participants prior to the training interventions. For the preinterventions lesson, participants were instructed to study the helping functions instructional manual and to explain it to themselves as well as they could before working through the associated exercises with the Lisp Tutor.

The syntax of the Lisp programming language is very different from that of English and can be difficult to verbalize. Prior to videotaping the preinterventions lesson, participants were asked to work through a set of warm-up exercises meant to provide practice in giving verbal protocols involving Lisp code. The two warm-up problems involved evaluating a Lisp function and writing a short Lisp function.

Phase 3: Instructional Interventions Phase

To investigate the causal nature of self-explanation and self-regulation strategies in learning and performance, we wanted to manipulate the application of specific strategies and examine the impact of using the strategies on student explanations and programming performance. The purpose of the instructional interventions phase was to explicitly train a group of participants (instructional group) to use the types of self-explanation and self-regulation strategies previously found to be used by high-performance students (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). Because exposure to the components of the training interventions may have an impact on learning and performance, a second training group served as a control. The instructional and control groups were equated for incoming performance levels on the basis of their programming performance scores from the preinterventions lesson. Based on our previous correlational findings, we assumed that this would also roughly balance the two training groups for incoming strategy use. The instructional and control groups received a sequence of training interventions equated for time on task and programming lesson materials. During the course of the interventions, both the instructional and control groups participated in activities based around a videotape of a student model using the trained self-explanation and self-regulation strategies. Both groups also studied and explained to themselves an instructional manual on a new programming topic. Only the instructional group participants, however, received explicit training in the application of self-explanation and self-regulation strategies.

The same student model was used for both groups. The model was presented using a videotape of a student studying the first two pages of the preinterventions manual. The student in the videotape performed the same self-explanation task

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Downloaded by [RMIT University Library] at 13:06 25 December 2017

that the participants performed in the preinterventions lesson. The model's lines were scripted to show the trained self-explanation and self-regulation strategies in use. The student model was a recruited university student of similar age group as the subject pool. The videotape had a running time of approximately 10 min.

Participants in both groups expected to participate in a variety of learning activities during the interventions sessions. During the introductory phase, participants had been informed that, in addition to learning programming and solving problems using the Lisp Tutor, there would be two sessions involving different types of learning activities. They were not informed that the additional learning activities would differ between participants.

The instructional group. The strategy training involved a structured one-to-one interaction between the experimenter and each participant in the instructional group. The intervention activities included (a) introducing and motivating the self-explanation and self-regulation strategies, (b) modeling the strategies using the student model on videotape, and (c) verifying a participant's ability to apply the strategies to instructional materials from a new programming lesson.

Instructional group Session 1. The first interventions session began with the experimenter introducing the self-explanation and self-regulation strategies and motivating the participant to use them. The approach involved moving from a general discussion of learning strategies, through top-level categories of self-explanation and self-regulation strategies, toward detailed discussion and modeling of specific strategies and their applications.

The experimenter informed the participant that the session would focus on study strategies for learning from instructional materials. We were interested in participants' incoming knowledge of learning strategies. The participant was asked to describe strategies that might be helpful in studying instructional materials such as the programming lesson manuals. All of the instructional group participants were familiar with the concept of learning strategies and were able to describe several strategies (e.g., asking themselves *why* questions, summarizing the main points of a lesson after reading through the materials).

The participant was also asked to watch the student model videotape and to comment on the types of learning strategies the student used. After being told that the student in the videotape would be studying the first two pages of the preinterventions lesson manual, the participant was asked to review these pages in the manual prior to watching the tape. The participant was then asked to watch the videotape. The participant was given a transcript of the video to aid in following the student model's explanations. The participant was also given the VCR controls in order to stop the video and ask questions at any point (although no participants did this).

Following this viewing, the experimenter introduced several self-explanation and self-regulation strategies. The experimenter explained that, although a variety

Downloaded by [RMIT University Library] at 13:06 25 December 2017

of strategies may be useful for studying, the training would focus on a specific set of strategies. The experimenter informed the participant that these strategies were chosen because prior research had found that high-performance students used these particular self-explanation and self-regulation strategies while studying instructional materials. The participant was given a handout listing the three top-level self-explanation strategies that would be covered in the session: (a) texts: identify and elaborate the relations between the main ideas, (b) examples: determine both the form and meaning of the Lisp code, and (c) texts and examples: connect the concepts in the texts and the examples.

For each top-level category, the experimenter presented the trained self-explanation and self-regulation strategies in the following manner:

Step 1. Describe the top-level category and explain its features and applications.
Step 2. Explain why it helps to use the strategies within this category.
Step 3. Present a self-interrogative method for applying strategies within the category incorporating issues of why/when/how to use the strategy.
Step 4. Discuss self-regulation strategies.

Step 3 provided a self-interrogative method involving a series of self-questions (presented on handouts). For example, for the strategy *Determine both the form and meaning of the Lisp code presented in the examples*, the following types of self-questions were used:

A general approach:
- Do I understand the definitions of the functions involved?
- Do I understand the structure of each construct?
- Do I understand how the code evaluates?

If you are looking at a particular piece of code:
- What is the purpose of this code?
- How does it achieve this purpose?
- Can I identify subparts of the code?
- What subproblems do these subparts solve?

The self-questions in Step 3 provided a format for discussing each self-explanation strategy, rather than as "formula steps" to be memorized as part of strategy application. For example, given the set of questions just mentioned, the experimenter presented possible self-explanation strategies for answering them: (a) evaluate the example code on concrete inputs and (b) focus on the form and function of the code in relation to the central concepts of a lesson.

The self-questions were also used to motivate the self-regulation strategies discussed in Step 4. One point of the self-regulation training was that participants should not simply determine if their self-explanations "made sense" but should also determine if their explanations answered these types of self-questions and prepared them for writing their own Lisp code (i.e., the goal task). Participants were trained (a) to monitor their comprehension, application of strategies, and progress in studying and preparing for the goal task and (b) to address and resolve comprehension failures.

The session ended with a second viewing of the student model videotape. The student in the videotape was used to provide a concrete model of the trained strategies. The experimenter narrated the student model's use of self-explanation and self-regulation strategies. This allowed the participant to observe and discuss the application of the trained learning strategies in a relevant and familiar learning situation.

Instructional group Session 2. The purpose of the second interventions session was to verify that the instructional group participants were able to apply the trained strategies on their own. The session began with a brief review. A wall diagram depicting the trained self-explanation and self-regulation strategies was shown. The experimenter used the diagram to review the strategies. The participant was given a few minutes to reread silently the handouts that had been presented during Session 1 and to ask questions. The experimenter then removed the handouts and wall diagram from view.

To determine how well participants were able to apply the trained strategies, participants were asked to study instructional materials on a new programming topic. The topic of iteration was chosen because its level of complexity was comparable to the postinterventions topic of recursion. The experimenter instructed the participant to study an instructional manual on iterative functions and to explain the materials to himself or herself as if preparing to solve associated programming problems on iteration. The experimenter asked the participant to use the trained self-explanation and self-regulation strategies.

The experimenter monitored the participant's strategy application while studying the instructional manual. We had anticipated the need to provide participants with guidance in using the trained strategies. To document the amount of intervention required by each participant, we developed a zone of proximal development prompting scheme.[1] (See Campione & Brown, 1990, for a discussion of this

[1]The zone of proximal development prompting scheme was based on a task analysis identifying a set of locations in the iteration manual where the trained strategies could be appropriately applied. If a participant was not producing any self-explanations or did not apply trained strategies at the task-analysis locations or during times of comprehension failure, the participant was to be prompted to self-explain the materials. The prompting scheme was designed to have the following levels of intervention:

Level 1	If:	The participant makes no self-explanations at a relevant point in the manual
	Then:	Ask the participant to self-explain the ideas presented in the materials.
Level 2	If:	The participant produces insufficient self-explanations at either Level 1 or at a relevant point in the manual
	Then:	Elicit further elaboration of the participant's own explanations.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Vygotskian concept and related prompting schemes.) However, all of the instructional group participants applied the self-explanation and self-regulation strategies satisfactorily and consistently throughout reading the iteration manual. Experimenter intervention was minimal and consisted primarily of prompting a participant to self-explain when the participant did not initiate any self-explanations.

After the participant completed studying the manual, he or she was informed that there would not be a set of iteration programming problems. Instead, there was a short debriefing period during which the participant was asked to respond orally to a set of questions concerning strategy application. The participant was told that the final session of the study would be similar to the preinterventions lesson on helping functions and that the trained self-explanation and self-regulation strategies should be used to study the instructional materials.

The control group. The control group interventions were meant to parallel the instructional group interventions without providing explicit strategy training. To control for the instructional group's exposure to the experimenter and to the materials during explicit strategy training, we gave the control group similar exposure within the context of a set of filler tasks. The filler tasks included discussion, memory recall tasks, and essay writing. These tasks were meant to engage the control participants with the experimenter and with the materials without teaching them specific self-explanation and self-regulation strategies or additional programming concepts.

Control group Session 1. The experimenter informed the participant that the session would be different from the typical programming lessons and instead focus on several experimental activities. To parallel the instructional group participants' discussion of learning strategies, the session began with a discussion about general programming issues. This activity allowed the experimenter and the participant to interact and the participant to express his or her own ideas about programming.

The participant was told that one of the session activities would involve watching a videotape of a student studying the first two pages of the preinterventions lesson manual. Whereas the instructional group participants reviewed the first two pages of the manual prior to watching the videotape, the control group participants both reviewed and performed a filler task using the materials. This task was meant to equate part of the time that the instructional group participants spent

Level 3	If:	The participant continues to produce insufficient self-explanations at Level 2
	Then:	Pose the self-questions from the handout in order to motivate the appropriate self-explanation strategy.
Level 4	If:	The participant continues to produce insufficient self-explanations at Level 3
	Then:	Model the appropriate self-explanations and monitoring aspects to the participant.

Because the intent was to monitor strategy applications, participants were not to be given feedback on either the content or the correctness of their self-explanations.

in explicit strategy training. The participant was given 20 min to memorize the text and example on pages 1 and 2 of the manual. The participant was allowed up to 15 min to reproduce the two pages of the manual as closely as possible.

The participant was then asked to watch the videotape. Control group participants were not given instructions to focus on the student model's learning strategies. Instead, the participant was informed that, following the videotape, there would be an activity involving the content of the videotape. The participant was given a transcript of the video to aid in following the student model's explanations. The participant was also given the VCR controls in order to stop the video and ask questions at any point (although no participants did this).

After watching the videotape, the experimenter instructed the participant to write a short summary of the video discussing the importance of helping functions in computer programming. Before writing the summary, the video was viewed a second time. This allowed control participants to prepare their essay and paralleled the second viewing by the instructional group. The experimenter watched the videotape with the participant but made no comments. The participant was then allowed 10 min to write a summary.

Control group Session 2. In the second control group session, the iteration study activity was the same as for the instructional group, except no directions were given to use particular learning strategies. The experimenter instructed the participant to study the iterative functions instructional manual and to explain the materials to himself or herself as if preparing to solve subsequent programming problems on iteration.

After the participant completed studying the manual, he or she was informed that there would not be a set of iteration programming problems. To provide an additional activity to equate time for the instructional group activities, the experimenter gave a memory recall task. The participant was given 10 min to memorize the text presented on page 1 of the iteration manual. Following this, the participant was allowed up to 10 min to reproduce the manual page as closely as possible.

Research on learning from text indicates that the anticipated criterial task influences a learner's goal and the nature of the study process (e.g., Kintsch, 1986; Schmalhofer & Glavanov, 1986). Thus, it was important not to change the learning goals of the control group to either memorization of instructional materials or preparation for writing essays on the content of a lesson. For this reason, we were careful to give participants in both groups specific instructions to study the instructional materials in Session 2 as if preparing to solve subsequent programming problems. We also emphasized to both groups that the learning activities during the interventions sessions would differ from those of the regular programming lessons. At the end of Session 2, the control group participants were informed that the final session of the study would not involve activities such as memory recall tasks and essay writing. The participant was told that the

Downloaded by [RMIT University Library] at 13:06 25 December 2017

final session would be similar to the preinterventions lesson on helping functions and that he or she would be studying materials on a new programming topic and solving corresponding problems.

Phase 4: Postinterventions Phase

The postinterventions phase involved a final programming lesson and was used to collect data on the explanations and programming performance of all participants subsequent to the training interventions. For the postinterventions lesson, participants were instructed to study the recursive functions instructional manual and to explain it to themselves as well as they could before working through the associated exercises with the Lisp Tutor.

Following the lesson, there was a debriefing period. All participants were asked to respond orally to a set of questions concerning the postinterventions topic of recursion, learning strategies, and participation in the study. The instructional group was asked additional questions about the trained strategies and their application.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

RESULTS

We examined the relations among strategy training, explanations, and programming performance. Specifically, we focused on the types of self-explanation and self-regulation strategies used by participants while studying the instructional materials and the participants' associated programming performance for the pre- and postinterventions lessons. These were investigated using quantitative and qualitative analyses of the participants' explanation protocols and programming performance measures. The central questions were:

- Was the improvement in programming performance of the instructional group greater than that of the control group from the pre- to postinterventions lessons?
- Were there corresponding improvements in the use of the trained self-explanation and self-regulation strategies by the instructional group subjects from the pre- to postinterventions lessons?

Programming Performance Measures

We first examined the programming performance measures collected via the Lisp Tutor for the pre- and postinterventions lessons. The Lisp Tutor recorded error information for each problem as part of a student performance record for the tutorial. The Lisp Tutor determines errors per problem by using an ideal programming model for each problem against which the number of incorrect code attempts made by participants at each point in the coding process are measured. (See Anderson, Conrad, & Corbett, 1989, for a discussion of error information in the Lisp Tutor.) The mean errors per problem for the preinterventions lesson ranged from 2.0 to 8.5, with the median at 4.2 and a mean of 4.9 (*SD* = 2.2). The mean errors per problem for the postinterventions lesson ranged from 1.4 to 11.2, with the median at 4.5 and a mean of 5.1 (*SD* = 2.4). As occurs in many educational settings, as the lessons progress, the level of difficulty increases. The postinterventions lesson on recursive functions was more difficult than the preinterventions lesson on helping functions, although the overall increase in mean errors per problem was not found to be significant.

The error data for the pre- and postinterventions lessons were used to characterize participant performance improvement.[2] The mean errors per problem for the instructional group decreased from 5.5 to 4.9 from pre- to postinterventions lessons. The mean errors per problem for the control group increased from 4.3 to 5.3 from pre- to postinterventions lessons. Considered individually, the change in mean errors per problem is not significant for either group. An analysis of variance (ANOVA) using a square root transformation on the error data, however, showed a significant interaction of Group (Instructional vs. Control) by Phases (Pre- vs. Postinterventions), $F(1, 22) = 4.70$, $p = .04$, $MS_E = .07$. Despite no overall change in mean errors per problem from the pre- to postinterventions lessons, the group receiving training in specific self-explanation and self-regulation strategies showed significantly greater performance gains than the control group.

Self-Explanation and Self-Regulation Strategies

We were interested in whether the instructional group's improvement in programming performance relative to the control group was related to an increased use of the self-explanation and self-regulation strategies on which they had been trained. To examine the participants' use of the trained strategies both before and after strategy training, we analyzed the explanation protocols collected as participants studied the instructional manuals for the pre- and postinterventions lessons.

The explanation protocols were coded using a scheme similar to that used in our prior self-explanation analyses (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). The verbal protocols for both lessons were transcribed and segmented

[2]In our earlier analyses in the domain of Lisp programming (Pirolli & Bielaczyc, 1989), we also examined performance measures based on other performance metrics. The results of the various performance measures were found to be consistent. Similarly, in the present study, we examined several performance measures. These included measures such as errors on first coding attempts at each step in the solution process and errors on all additional coding attempts at each step. We also examined combined errors from the first list recursion problem and the first number recursion problem from the postinterventions lessons. As in our earlier analyses, the results were consistent. The analyses on all error measures indicated greater performance gains for the instructional group relative to the control group. The mean errors per problem (first + additional coding attempts) for each lesson were assumed in the present analysis to be a good general measure of performance.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

into *elaborations*. We define an elaboration as a pause-bounded utterance that is not a first reading of the texts or examples or a conversation with the experimenter. Each elaboration was placed in one of the following categories:

Domain elaborations: statements about and related to programming, for example: "The helping functions do simplify a lot." "And its value would change each time, because there would still be a list left, which would be the *cdr* of *lis*." "Then, I guess you can go back and start evaluating the functions."
Monitor elaborations: statements about one's state of understanding or about oneself as a learner, for example: "Okay, that makes more sense now." "I'm not quite sure what that is." "I like examples 'cause I understand them better than words."
Strategy elaborations: statements about a planned approach to explaining the materials, for example: "I'm just gonna look at things I'm not really sure about and just read on." "Wait a minute here, well I'm going to use an example of my own using this."
Activity elaborations: statements about the instructional task or materials, for example: "It would have been nice if they'd have started with an example." "This paragraph is too long."
Reread elaborations: statements that are rereadings of the instructional texts or examples.
Incomplete elaborations: statements that are incomplete phrases or utterances, for example: "And then it'd put . . ."

An interrater reliability measure was obtained by having two coders familiar with Lisp programming independently code one of the longer verbal protocols. The coders agreed on 83% of their initial coding assignments; differences in coding were generally easily resolved.

Our analyses of the explanation protocols focused on participants' use of the specific strategies that the instructional group had been trained to apply. In particular, we examined the use of substrategies within each of the following top-level categories:

1. self-explanation strategies
   a. texts: identify and elaborate the relations between the main ideas
   b. examples: determine both the form and meaning of the Lisp code
   c. texts and examples: connect the concepts in the texts and the examples
2. self-regulation strategies
   a. monitoring comprehension and learning activities
   b. clarifying and addressing comprehension failures.

The domain elaborations were analyzed for the types of self-explanation strategies that had formed the basis of the strategy instruction. The monitor and strategy elaborations were analyzed for the trained self-regulation strategies. The analyses focus on changes in the use of strategies by each group (Group) from the pre-

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Downloaded by [RMIT University Library] at 13:06 25 December 2017

to postinterventions lessons (Phases). The analyses are based on logarithmic transformations of the data; however, the untransformed means are reported. The tables illustrate the mean number of times a strategy was used by the instructional and control group participants.

In the strategy analyses presented next, a main effect of phases was consistently found, $p < .005$. The effect reflects an overall increase in strategy use. This was expected due to (a) the increased difficulty of the postinterventions topic of recursive functions over the preinterventions topic of helping functions and (b) the impact of the training interventions on both groups. The more difficult materials impose a higher cognitive demand, resulting in an increased effort in understanding and explaining by both groups. Although both groups increase in their use of the trained strategies, our main interest is in whether the instructional group shows a greater increase relative to the control group.

Self-Explanation Strategy: Identify and Elaborate the Relations Between the Main Ideas in the Text

Strategy training for the strategy type *Identify and elaborate the relations between the main ideas introduced in the text* emphasized locating the central concepts in the text and constructing an interrelated model of the main lesson topic by elaborating the concepts. The domain elaborations generated while studying text pages were analyzed for elaborations of the main ideas of a given lesson. The following are examples of main idea elaborations:

> A central concept in the preinterventions lesson on helping functions is that helping functions simplify complex code: "So, you can break a problem down into easier parts to solve and then code the subparts using helping functions."
>
> A central concept in the postinterventions lesson on recursion is the recursive call: "The recursive call, meaning it jumps into its recursive loop."

Two analyses were performed in examining the use of the self-explanation strategy of identifying and elaborating the main ideas of the text. In the first analysis, the overall number of main idea elaborations made by instructional group and control group participants for the pre- and postinterventions lessons were compared (Table 1). The within-group gains for both groups were significant: instructional group, $F(1, 10) = 77.68$, $p < .001$, $MS_E = .08$; control group, $F(1, 12) = 13.46$, $p < .01$, $MS_E = .08$. The increase shown by the instructional group, however, was significantly greater relative to that of the control group. There was a significant Group $\times$ Phases interaction for main idea elaborations, $F(1, 22) = 16.10$, $p = .0006$, $MS_E = .08$. The instructional group also generated significantly more main idea elaborations than the control group while studying the postinterventions lesson manual, $F(1, 22) = 7.49$, $p = .01$, $MS_E = .14$.

TABLE 1
Text Strategies: Identifying and Elaborating the Main Ideas

	Total Main Idea Elaborations		*Coverage of Main Ideas*	
Group	*Preinterventions*	*Postinterventions*	*Preinterventions*	*Postinterventions*
Instructional	2.5	35.0	2.4	14.1
Control	5.5	13.8	3.9	9.1

The instructional manuals for the pre- and postinterventions lessons introduced several central concepts associated with the subject matter of each lesson. Another perspective on the application of the main ideas strategy concerns a participant's "coverage" of the concepts introduced in the texts throughout the manual (pages 1, 3, and 5). In the second analysis, we examined the distinct number of main ideas that were elaborated by participants while studying the texts. For each of the main ideas introduced in the manual, it was determined whether a participant did or did not elaborate that main idea (Table 1). The within-group gains in the number of main ideas for which participants generated self-explanations were significant for both groups: instructional group, $F(1, 10) = 54.50$, $p < .001$, $MS_E = .06$; control group, $F(1, 12) = 15.07$, $p < .01$, $MS_E = .06$. Again, the increase was significantly greater for the instructional group relative to the control group. There was a significant Group $\times$ Phases interaction for the coverage of main ideas, $F(1, 22) = 7.93$, $p = .01$, $MS_E = .06$.

These analyses indicate that not only did instructional participants show a significantly greater increase than control participants in their overall application of the strategy of elaborating the main ideas while studying the text, but they were also applying the strategy to a greater number of the main ideas introduced in the texts.

Self-Explanation Strategy: Determine Both the Form and Meaning of Example Code

The training for the self-explanation strategy *Determine both the form and meaning of example code* emphasized the following two approaches to self-explaining example code: (a) evaluating the code using concrete values as inputs to the given Lisp functions and (b) elaborating the form and function of the code in relation to the main lesson topic. The elaboration protocols were analyzed to determine whether domain elaborations reflecting these substrategies were produced by the participants while reading the examples given in the pre- and postinterventions manuals.

In the first analysis, we examined whether participants evaluated the code in the given examples using concrete inputs. The strategy was not widely applied during the preinterventions lesson. This made it inappropriate to use an ANOVA,

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Downloaded by [RMIT University Library] at 13:06 25 December 2017

because the homogeneity of variance assumption would have been violated. Instead, a comparison of change scores was used. Based on the change scores from pre- to postinterventions lessons, we found that the instructional group participants increased their application of the strategy significantly more than the control group participants: $t(22) = 3.07$, $p = .003$, $MS_E = .007$. The instructional group participants were using the strategy of stepping through how an example program operates by evaluating the code on specific inputs more often than the control group participants.

The self-explanations generated by participants while studying the example code varied from elaborating the syntax and the definitions of basic Lisp functions contained in the example to elaborating how parts of the example code exemplified the main lesson topic. For the second analysis, the instructional and control groups were compared for example elaborations focusing on the main topic of a given lesson. For the preinterventions lesson, we examined elaborations focusing on how the code related to the topic of helping functions; for the postinterventions lesson, we examined elaborations focusing on recursion. For example, the following are elaborations made while studying the examples of the postinterventions lesson that do not relate to the topic of recursion: "Then it will *cons* the value, the *a*, into the list." "*Defun* the name and then the element that's being put in is a list." The following are examples of recursion-related elaborations: "Then *cons* that, you're making a list with *a*, but that doesn't start until later when we hit *nil*." "So then it's going back to the function you're just defining."

The results of the analysis are illustrated in Table 2. Within each group, subjects showed significant increases in generating example elaborations focused on the main lesson topic: instructional group, $F(1, 10) = 21.33$, $p < .001$, $MS_E = .09$, control group, $F(1, 12) = 18.75$, $p < .001$, $MS_E = .04$. Although the increases in the mean number of example elaborations focusing on the lesson topics for the pre- and postinterventions lessons follow the same general trend as the other strategy analyses, no significant Group × Phases interaction was found, $F(1, 22) = 2.90$, $p = .10$, $MS_E = .06$. The gain for the instructional group compared with the control group for this strategy falls short of a reliable difference. The instructional group was found to generate significantly more elaborations of the main lesson topic than the control group while studying the example pages of the postinterventions lesson manual, $F(1, 22) = 4.69$, $p = .04$, $MS_E = .09$.

TABLE 2
Example Strategies: Determining Both the Form and Meaning of Example Code

	Example Evaluations Using Concrete Inputs		*Elaborations Focused on Main Lesson Topic*	
Group	*Preinterventions*	*Postinterventions*	*Preinterventions*	*Postinterventions*
Instructional	0	1.7	8.1	35.5
Control	.1	.8	8.2	17.1

Self-Explanation Strategy: Connect the Concepts Presented in the Texts and Examples

Training in the self-explanation strategy of *Connect the concepts presented in the texts and examples* focused on the following types of connections: (a) text–example connections—clarifying the meanings of abstract concepts presented in the text by explicitly connecting the concepts to the ways in which they are applied in the given examples, (b) example–example connections—attaining a better understanding of how example code works by relating the code to other examples in the instructional materials, and (c) external-to-lesson connections—integrating new and prior knowledge by relating the lesson concepts to concepts external to those in the instructional manual. The domain elaborations were analyzed for each of these substrategies. Again, we analyzed change scores because the strategies were not widely applied during the preinterventions lesson.

In the first analysis, the explanation protocols for the pre- and postinterventions lessons were analyzed to determine whether the subjects explicitly identified relations between the concepts and principles stated in the text and their instantiations in the examples. The following are sample elaborations connecting the concepts presented in the texts and the examples:

While studying an example in the helping functions manual, the participant identifies which of several Lisp functions are the *helping functions*—a concept introduced in the text of the previous page: "Oh, *negative root* along with *positive root* are the helping functions making up the *quadratic function*."

While studying the recursion manual, the participant reads in the text: "When a recursive function gets a terminating value as input, it returns a value that does not involve a recursive call," and connects the concept of Value that does not involve a recursive call to the example code *(cond ((null lis) nil)*: "So, here you mean it will return the nil."

The increases in the mean number of text–example connection elaborations are shown in Table 3. There was a significant difference in change scores from the pre- to postinterventions lessons, $t(22) = 2.33$, $p = .01$, $MS_E = .03$. The

TABLE 3
Text and Example Strategy: Connecting Concepts in the Texts and Examples

	Text–Example Connections		*Example–Example Connections*		*External-to-Lesson Connections*	
Group	*Pre-interventions*	*Post-interventions*	*Pre-interventions*	*Post-interventions*	*Pre-interventions*	*Post-interventions*
Instructional	1.5	21.1	.6	9.6	.09	1.64
Control	1.8	7.8	1.3	2.9	.54	.85

Downloaded by [RMIT University Library] at 13:06 25 December 2017

instructional group showed a greater increase than the control group in the strategy of connecting the text concepts to their instantiations in the examples.

In the second analysis, the pre- and postintervention explanation protocols were analyzed for elaborations connecting features of an example presented on page 2, 4, or 6 to example code on another page (Table 3). For example–example connection elaborations, there was a significant difference in change scores from the pre- to postinterventions lessons, $t(22) = 2.40$, $p = .01$, $MS_E = .04$. The instructional group showed a greater increase than the control group in using the strategy of relating features of different examples within a lesson.

In the third analysis, the explanation protocols were analyzed to determine whether the participants related the central concepts of the lesson to concepts external to the lesson manual, for example:

The participant reads the term *recursive* in the text of the postinterventions manual and uses knowledge of the prefix *re-* in order to gain a deeper understanding of term's meaning: "Recursive, *re-* is usually doing things over and over again."

The participant reads the description of the *recursive relation* in the text of the postinterventions manual and is reminded of functions operating on functions in the context of mathematics: "So it's sort of like *f* of *f(x)*. I get it. It goes back to functions stuff."

The mean number of external-to-lesson connections for each group is shown in Table 3. There was a significant difference in change scores from the pre- to postinterventions lessons: $t(22) = 2.34$, $p = .01$, $MS_E = .01$. Not only did the instructional group show a greater increase in the use of strategies connecting text concepts and example features found within the lesson materials than the control group, but they also showed a greater increase in the strategy of connecting lesson concepts to topics external to the lesson materials.

Self-Regulation Strategy: Monitoring Comprehension and Learning Activities

Strategy training focused on the following two main classes of self-regulation strategies: (a) monitoring one's comprehension, strategy application, and learning progress and (b) methods of clarifying one's understanding and addressing comprehension failures. Regarding the first class, information on these types of self-regulation strategies was obtained by dividing elaborations in the monitoring category into two subcategories: (a) general monitoring elaborations and (b) comprehension failure monitoring elaborations. General monitoring elaborations reveal strategies such as monitoring positive states of comprehension: "Oh, right, I remember that." After reading a sentence in the text and stating, "OK," checking the steps of one's explanation activities: "Wait, that's, that's it." "No, I didn't mean to say that." and using awareness of one's learning styles or characteristics

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Downloaded by [RMIT University Library] at 13:06 25 December 2017

to guide learning activities: "I like examples 'cause I understand them better than words."

Comprehension failure monitoring elaborations refer to negative statements about one's state of knowledge or comprehension of a concept, for example: "I don't get this at all." "What is this *cdr* thing again?"

The explanation protocols of the instructional and control group participants were analyzed for the general monitoring and comprehension failure elaboration subcategories. Analyses of the general monitoring elaborations are presented here; the comprehension failure elaborations are discussed next.

We thought that the effectiveness of the self-regulation strategies would depend on the content of the explanations being monitored. Separate analyses were performed for general monitoring elaborations made while studying the text pages and while studying the example pages. While studying the text pages, participants' explanations typically remained focused on the central concepts of the lesson. The results of analyzing the overall general monitoring of participants while generating these explanations are reported in Table 4. The within-group gains for both groups were significant: instructional group, $F(1, 10) = 44.19, p < .001, MS_E = .07$; control group, $F(1, 12) = 8.9, p < .01, MS_E = .07$. The Group × Phases interaction was also found to be significant, $F(1, 22) = 8.23$, $p = .009$, $MS_E = .07$. The increases in general monitoring strategies while studying texts were significant for both groups, with the increase for the instructional group significantly greater relative to the control group. The instructional group also used significantly more general monitoring strategies than the control group while studying the text pages of the postinterventions lesson manual, $F(1, 22) = 12.49$, $p = .002, MS_E = .1$. While studying the example pages, participants' explanations typically varied from elaborating features of basic Lisp functions to elaborating the form and function of the code in relation to the main lesson topic. The results of analyzing the overall general monitoring of participants while generating explanations of the examples are reported in Table 4. Within each group, the participants showed significant increases in overall general monitoring: instructional group, $F(1, 10) = 27.52, p = < .001, MS_E = .05$; control group, $F(1, 12) = 6.31$, $p < .03$, $MS_E = .07$. The overall amount of general monitoring by the

TABLE 4
Self-Regulation Strategy: Monitoring Comprehension and Learning Activities

	General Monitoring of Text Pages		*General Monitoring of Example Pages*		*General Monitoring of Example Pages (Topic Related)*	
Group	*Pre-interventions*	*Post-interventions*	*Pre-interventions*	*Post-interventions*	*Pre-interventions*	*Post-interventions*
Instructional	7.2	30.6	13.7	41.6	3.0	17.1
Control	7.2	11.8	14.7	20.5	4.1	9.8

instructional group was also significantly more than the control group while studying the example pages of the postinterventions lesson manual, $F(1, 22) = 8.14$, $p = .009$, $MS_E = .05$. The Group × Phases interaction for this measure was not found to be significant, $F(1, 22) = 1.86$, $p = .19$, $MS_E = .06$. When we examined a more restricted set of the general monitoring elaborations generated while studying the examples, however, those that specifically monitored domain elaborations of the main lesson topic, the results were found to follow the same general trend as the other strategy analyses (Table 4). The Group × Phases interaction for this restricted set fell short of significance, $F(1, 22) = 3.73$, $p = .06$, $MS_E = .06$. The combination of this trend with the significant gain of the instructional group relative to the control group for general monitoring of text suggests that the content of the self-explanations that are being monitored may play a role in the effectiveness of the strategy.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Self-Regulation Strategy: Clarifying and Addressing Comprehension Failures

In analyzing the number of comprehension failure elaborations made during the pre- and postinterventions lessons, the same general trend in increases appears (Table 5). These increases may reflect an increased application of self-regulation strategies. They may also reflect that, as the instructional materials increase in difficulty, participants experienced an increased number of comprehension failures. We felt that the number of comprehension failure elaborations was not in itself an informative measure in the context of examining the trained self-regulation strategies. We did not train participants to experience comprehension failures. Rather, we trained them in methods of clarifying their understanding and resolving comprehension failures.

To analyze the use of self-regulation strategies for clarifying understanding and addressing comprehension failures, we examined the monitor and strategy elaborations for evidence of the following types of substrategies:

1. Reviewing at the end of a page and assessing comprehension difficulties.

During training, participants had been informed that, after studying a given page, it was useful to review the contents of the page to assess what was under-

TABLE 5
Comprehension Failure Elaborations

	Comprehension Failures on Text Pages		*Comprehension Failures on Example Pages*	
Group	*Preinterventions*	*Postinterventions*	*Preinterventions*	*Postinterventions*
Instructional	2.3	10.8	3.4	14.1
Control	.92	3.1	1.9	8.0

Downloaded by [RMIT University Library] at 13:06 25 December 2017

stood and not understood before continuing to subsequent pages. For the analysis, we examined whether a participant reviewed a given page and self-assessed his or her understanding on reaching the end of a page in the instructional manual. This analysis was independent of whether other elaborations had been generated in the course of studying a given page. The following is an example:

The participant finishes reading and elaborating the second example in the postinterventions manual, starts to turn the page, then decides that he should look back over the example and summarize his understanding of how the example code operates. After he elaborates his understanding, he concludes: "The only thing I'm not clear on is exactly how the answer comes about with the *car* of *car* of the *lis*."

2. Clarifying comprehension failures.

Participants had been trained to clarify what was understood and not understood at points of comprehension failure (i.e., to determine the source of difficulty as opposed to simply expressing a comprehension failure). For the analysis, we examined the number of comprehension failures that were accompanied by clarifications. The following are examples of this strategy:

As the participant is reading and elaborating the example code, a comprehension failure occurs along with the following clarification elaborations: "Okay, I understand the first part and the second part. It's the, the little changing thing that I don't understand."

As the participant is working through the evaluation of the code in an example, a comprehension failure occurs along with the following clarification elaborations: "Then, you're going back up and you're adding *a* to that. That, I understand that. What I don't understand is how you go from, when it finally gets to just *nil*, how you know to go back to *c*."

3. Go on and come back: Specifically returning to earlier pages to resolve comprehension failures.

Self-regulation training for comprehension failures included emphasizing the importance of making and following through on a plan for resolving comprehension failures. For the present analysis, we examined a particular resolution strategy: Go on and come back. The strategy is to improve one's understanding by continuing on through the materials and then returning to the point of comprehension failure. It should be noted that more participants state this plan than fully execute it. We examined points in studying the lesson manual at which participants executed the plan by specifically returning to clarify an earlier misunderstanding, for example:

The participant makes a plan to continue and return later to a given page where comprehension failures occurred. After studying several subsequent pages of the lesson manual, the participant states: "Okay, maybe I should go back and read over this other stuff, having reached a little bit clearer understanding of the vocabulary." The participant returns to the earlier page and elaborates the terminology. The participant experiences difficulty with the concepts presented in a given page of text, reads through to the end of the page and states: "I'll go on and come back." The participant elaborates the example on the next page and turns to the subsequent page of text, then stops and decides to return to the previous text page where the comprehension difficulties occurred: "Like I just got to, wait a minute, because I want to go back." The participant returns to the earlier text page and, after rereading and elaborating, states: "Just wanted to clarify that."

The results of the analysis are illustrated in Table 6. Although the instructional group participants showed a significant within-group increase in their use of self-regulation strategies for clarifying and addressing comprehension failures, the control group did not show a similar within-group increase: instructional group, $F(1, 11) = 30.83$, $p < .001$, $MS_E = .03$; control group, $F(1, 12) = 1.75$, $p = .2$, $MS_E = .03$. The Group × Phases interaction was found to be significant, $F(1, 22) = 10.71$, $p = .003$, $MS_E = .03$. The amount of self-regulation strategies applied while studying the postintervention lesson manual was also significantly more for the instructional group than the control group, $F(1, 22) = 9.65$, $p = .005$, $MS_E = .06$. The instructional group showed significant increases compared with the control group in their application of self-regulation strategies for clarifying understanding and addressing comprehension failures.

Summary of the Explanation and Performance Analyses

In summary, the instructional group showed significantly greater increases than the control group in the use of the trained self-explanation and self-regulation strategies from the pre- to postinterventions lessons. Furthermore, increased strategy application was accompanied by significantly greater performance gains for the instructional group compared with the control group. The analyses also suggest a materials and interventions effect for both groups. Regarding the materials effect, the complexity of the subject matter increases from pre- to postinterventions lessons. As the difficulty level of the lessons increases, the expected effect on performance is an increase in errors per problem; the expected effect on studying materials is an increase in the application of self-explanation and self-regulation strategies. Regarding the interventions effect, because both groups received relevant strategy training between the two lessons, the expected effect

TABLE 6
Self-Regulation Strategy: Clarifying and Addressing Comprehension Failures

	Strategies for Clarifying Understanding and Addressing Comprehension Failures	
Group	*Preinterventions*	*Postinterventions*
Instructional	1.1	4.2
Control	1.2	1.5

Downloaded by [RMIT University Library] at 13:06 25 December 2017

is an increase in strategy application across lessons by both groups. Relative to the group receiving only implicit strategy training, however, it was the group receiving explicit training that showed not only a significant gain in strategy application but also an accompanied significant gain in performance.

DISCUSSION

The main purpose of this study was to provide experimental support for the hypothesis that specific self-explanation and self-regulation strategies contribute to learning and performance. To achieve the necessary experimental manipulations, we developed instructional interventions to communicate these learning strategies. The instructional interventions included explicitly telling participants about the purpose, form, and context of use of the strategies, demonstrating the strategies through explicit examples exhibited by a student model, and providing guided practice on the application of the strategies.

Use of the trained self-explanation and self-regulation strategies led to improved cognitive skill acquisition and programming performance. Instructional group participants received training on specific strategies between programming lessons and showed greater improvement in applying these strategies across the lessons than control group participants. This increased application of the trained strategies by the instructional participants was accompanied by greater performance gains compared with the control participants.

Previous research found interesting correlations between students' explanation strategies and their subsequent problem-solving performance (Chi et al., 1989; Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994). The present study strengthens those results by showing experimentally that training students in the types of self-explanation and self-regulation strategies found to be used by high performers can improve students' study strategies, which in turn can improve learning of a problem-solving skill. Although both groups were exposed to a model of effective learning strategies, the instructional group was explicitly trained to apply specific self-explanation and self-regulation strategies. Mere exposure to the effective strategies leads to some improvement, but of an order of magnitude less than the practice provided in the instructional condition. The application of these specific strategies appears to affect learning from instructional materials and solving associated programming problems.

In attempting to get closure on issues raised by earlier research, the present investigation raises new issues. One issue concerns why the application of particular self-explanation and self-regulation strategies contributes to learning from instructional materials prior to problem solving. In general, we expect that the explanations generated by using effective self-explanation and self-regulation strategies result in well-integrated, coherent representations. Possible implications of such declarative representations for subsequent problem solving include (a)

Downloaded by [RMIT University Library] at 13:06 25 December 2017

concepts and procedures are more memorable because multiple retrieval cues exist, (b) specific memories are created that match the conditions of use for a problem-solving skill (e.g., goals achieved by the skill, contexts in which the skill applies), and (c) inferences about and applications of a skill in new contexts are more easily accomplished because the self-explanations make explicit the underlying rationale for the skill.

In the domain of programming, students need not only to learn and recall the exact syntax of programming commands but also to make inferences, abstract rules and procedures, and determine how to effectively combine commands in order to gain an understanding of the instructional materials appropriate for subsequent programming tasks. We anticipate that the explanations generated by using the trained self-explanation and self-regulation strategies result in declarative representations well suited to the target task of designing and writing one's own computer programs. Our interventions were designed to communicate self-explanation and self-regulation strategies that our prior research in the domain of programming (Pirolli & Bielaczyc, 1989; Pirolli & Recker, 1994) suggested would improve learning and performance. We believe that these particular strategies are effective for the following reasons:

1. Self-explanation strategy: Identify and elaborate the relations between the main ideas of text. The production of elaborations associated with this self-explanation strategy roughly corresponds to constructing chunks of declarative knowledge concerned with program structure, design, and code evaluation. This knowledge should be particularly useful during subsequent problem-solving exercises in which learners are required to interpret a description of a desired Lisp function and then design and code the actual function. Elaborating the interrelations among concepts while studying instructional materials should provide a more comprehensive knowledge base than isolated chunks of declarative knowledge. During subsequent problem solving, well-elaborated declarative representations should aid in the recall of relevant terminology and required components of a given function type, the development of the appropriate structure for a function, and heuristics for design.

2. Self-explanation strategy: Determine both the form and meaning of example code. The example solutions typically found in programming instruction present a concrete set of actions or completed code with little explanation for the underlying rationale. Generating explanations that identify the processes and rationale underlying those actions or code is important, because writing code can be characterized as a task in which goals to produce certain processes must be instantiated into concrete code. The declarative links established between program form and meaning while explaining examples may later be followed when meaningful intentions have to be translated into code.

3. Self-explanation strategy: Connect the concepts presented in the texts and examples. Although many learners are able to understand at some level the meaning of a concept when it is explained in the text, a more refined level of

Downloaded by [RMIT University Library] at 13:06 25 December 2017

understanding is obtained in identifying the connections between descriptions and instantiations. In terms of cognitive processing, having greater amounts of declarative knowledge about concepts in the domain connected to the representations of the example increases the probability of retrieving relevant information during problem-solving tasks. By elaborating connections between examples, a learner may be better able to understand the use or significance of particular pieces of code. Such an understanding is important for later programming tasks requiring the generation of code. By integrating lesson concepts with topics external to the lesson, a student is able to create meaningful links between to-be-learned concepts and concepts with which the student is already familiar.

4. Self-regulation strategies. In addition to developing a repertoire of knowledge acquisition strategies, it is important for learners to be able to plan, monitor, and evaluate their comprehension and strategy use. Through comprehension monitoring, a learner may determine whether the explanations he or she is generating help achieve an understanding of the instructional materials suited to the criterial task of programming. An awareness of the strengths and weaknesses in one's understanding can aid in determining which types of study strategies to apply and in determining the effectiveness of their application.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Another issue raised by this study concerns how strategies interact with other components of the learning process in order to result in effective self-explanations. The study confirms that the trained self-explanation and self-regulation strategies play an important role in the learning process. Our conjecture is, however, that the effectiveness of the strategies themselves varies depending on several factors, including (a) a learner's prior knowledge (both domain-general and domain-specific knowledge), (b) the quality of the content of the explanation generated from applying a particular strategy, (c) the cohesiveness or clarity of the materials being studied, and (d) the state of one's evolving understanding. That is, given two students who are both applying the same self-explanation and self-regulation strategies, we would expect the effectiveness of the resultant self-explanations to depend on factors such as those listed here. For example, given similar strategies, a student whose evolving model of recursion is more robust at a given point in the materials than another student's would be expected to be able to generate higher quality explanations at that point than the other student. Furthermore, we would expect the resultant declarative representation to become part of a more advanced knowledge base.

The interaction of self-explanation and self-regulation strategies with other aspects of the learning environment provides an agenda for future investigations. This study provides an important step toward future investigations of this type. To examine the interaction between learning strategies and other components of the learning process, it is necessary to articulate a relevant set of learning strategies and to show that they play a contributing role in learning. In this study, we tested our articulation of a specific set of strategies by communicating the strategies to students and explicitly training them to apply such strategies. The results of the strategy training provide evidence that the trained self-explanation and self-regulation strategies play a contributing role in the acquisition of cognitive skills. By building on the results of prior research and providing a basis for future investigations, this study contributes to our understanding of the role of student-generated explanations and metacognition in learning from instructional materials.

ACKNOWLEDGMENTS

Peter Pirolli is now at Xerox PARC.

Portions of this research have been presented at meetings of the American Educational Research Association, April 1991, and the International Conference on the Learning Sciences, August 1991. The research and preparation of this article were supported by a Spencer Dissertation fellowship and a University of California Regents research fellowship to Katerine Bielaczyc, a grant from the National Science Foundation (IRI–9001233) to Peter Pirolli, and a grant from NIH (NICHD–06864) to Ann Brown.

We thank Mimi Recker and Michael Reynolds for their contributions at various stages of this research and Joseph Campione and Allan Collins for extensive comments on earlier drafts.

REFERENCES

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J. R. (1987). Skill acquisition: The compilation of weak-method problem solutions. *Psychological Review, 94,* 192–210.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13,* 467–505.

Anderson, J. R., Corbett, A., & Reiser, B. P. (1987). *Essential LISP.* Reading, MA: Addison-Wesley.

Bereiter, C., & Scardamalia, M. (1989). Intentional learning as a goal of instruction. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 361–392). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Baker, L., & Brown, A. L. (1984a). Cognitive monitoring in reading. In J. Flood (Ed.), *Understanding reading comprehension* (pp. 21–44). Newark, DE: International Reading Association.

Baker, L., & Brown, A. L. (1984b). Metacognitive skills and reading. In D. Pearson, M. L. Kamil, R. Barr, & P. Mosenthal (Eds.), *Handbook of reading research* (pp. 353–394). New York: Longman.

Brown, A. L. (1980). Metacognitive development and reading. In R. J. Spiro, B. Bruce, & W. Brewer (Eds.), *Theoretical issues in reading comprehension* (pp. 453–481). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Brown, A. L., Armbruster, B. B., & Baker, L. (1985). The role of metacognition in reading and studying. In J. Orasanu (Ed.), *Reading comprehension: From research to practice* (pp. 49–75). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, remembering, and understanding. In J. H. Flavell & E. M. Markman (Eds.), *Handbook of child psychology: Vol. 3. Cognitive development* (4th ed., pp. 77–166). New York: Wiley.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

Brown, A. L., Campione, J. C., & Day, J. D. (1981). Learning to learn: On training students to learn from texts. *Educational Researcher, 10*(2), 14–21.

Campione, J. C., & Armbruster, B. B. (1984). An analysis of the outcomes and implications of intervention research. In H. Mandl, N. Stein, & T. Trabasso (Eds.), *Learning and comprehension of texts* (pp. 287–304). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Campione, J. C., & Brown, A. L. (1990). Guided learning and transfer: Implications for assessment. In N. Frederiksen, R. Glaser, A. Lesgold, & M. G. Shafto (Eds.), *Diagnostic monitoring* (pp. 141–172). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, 145–182.

Chipman, S., Segal, J., & Glaser, R. (1985). *Thinking and learning skills: Current research and open questions*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Collins, A., Brown, J. S., & Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453–494). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Eylon, B. S., & Helfman, J. (1989). *The effect of declarative knowledge, worked examples, and strategies on problem-solving schemes in physics*. Unpublished manuscript.

Ferguson-Hessler, M. G. M., & de Jong, T. (1990). Studying physics texts: Differences in study processes between good and poor solvers. *Cognition and Instruction, 7*, 41–54.

Kieras, D. E. (1985). *Improving the comprehensibility of a simulated technical manual* (Tech. Rep. No. 20 TR–85/ONR–20). University of Michigan.

Kintsch, W. (1986). Learning from text. *Cognition and Instruction, 3*, 87–108.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1–55). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Pirolli, P. (1987). A model of purpose-driven analogy and skill acquisition in programming. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 609–621). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Pirolli, P. L., & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skill. *Canadian Journal of Psychology, 39*, 240–272.

Pirolli, P., & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society* (pp. 450–457). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Pirolli, P., & Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction, 12*, 235–275.

Reder, L., Charney, D., & Morgan, K. (1986). The role of elaborations in learning a skill from instructional text. *Memory and Cognition, 14*, 64–78.

Reiser, B. J., Anderson, J. R., & Farrell, R. G. (1985). Dynamic student modelling in an intelligent tutor for Lisp programming. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 8–14). Los Altos, CA: Morgan-Kaufman.

Ross, B. H. (1980). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology, 16*, 371–416.

Schmalhofer, F., & Glavanov, D. (1986). Three components of understanding a programmer's manual: Verbatim, propositional, and situational representations. *Journal of Memory and Language, 25*, 279–294.

vanDijk, T. A., & Kintsch, W. (1983). *Strategies of discourse representation.* New York: Academic.

Winston, P. H., & Horn, B. K. P. (1981). *LISP*. Reading, MA: Addison-Wesley.

Downloaded by [RMIT University Library] at 13:06 25 December 2017

APPENDIX

The Postinterventions Lesson Manual on Recursive Functions (Pages 1 and 2)

Recursive Functions
A *recursive function* solves a complex problem by using itself as a helping function to solve part of the problem. When a recursive function calls itself as a helping function, we say that it is making a *recursive call.*

The Structure of a Recursive Function
The definition of a recursive function contains a cond structure. The cond structure consists of *terminating cases* and *recursive cases.* A terminating case performs a test and returns a value that does not require a recursive call. The input value that a terminating case tests for is called the *terminating value* of the input. A recursive case performs a test and returns the value of a recursive relation. A *recursive relation* is the relation between the result of a recursive call and the result of the recursive function.

On the next page is an example of recursive function.
Please try to understand the example function and explain it to yourself out loud.

1

An Example of a Recursive Function
EXAMPLE: *carlist*
Carlist is a function that takes an arbitrary list, L, that contains other lists as its elements, and returns a list containing the first element of each list in L.

For example: (carlist '((a (b)) (c d) ((e) f))) = (a c (e))

The recursive definition of carlist is:

```
(defun carlist (lis)
      (cond   ((null lis) nil)
              (t (cons (car (car lis)) (carlist (cdr lis))))))
```

2

Downloaded by [RMIT University Library] at 13:06 25 December 2017