The Effects of Self-Explanation on Studying Examples and Solving Problems

William A. Sandoval
Institute for the Learning Sciences and
School of Education & Social Policy
Northwestern University
Evanston, Illinois 60208
sandoval@ils.nwu.edu

J. Gregory Trafton
Naval Research Laboratory
Code 5513
4555 Overlook Ave. S.W.
Washington, DC 20375-5337
trafton@itd.nrl.navy.mil

Brian J. Reiser
Institute for the Learning Sciences and
School of Education & Social Policy
Northwestern University
Evanston, Illinois 60208
reiser@ils.nwu.edu

Abstract

Examples play a critical role in guiding the acquisition of cognitive skills. We have argued that students need to apply the knowledge gathered from studying examples to solve analogous problems for that knowledge to be effective. There is a tradeoff between the active nature of constructing solutions and the facilitating effect of guiding problem solving with a worked example. The present study examined the impact of self-explanations on the effectiveness of examples in guiding later problem solving. We found that within a learning environment which provided direct support for the self-explanation of worked examples, such study could be as effective as direct problem solving practice.

Introduction

There has been much attention focused on the role of examples in acquiring new skills. Examples clearly play a critical role in guiding learning (Sweller, 1988). Students strongly focus on examples in instructional material, and the ways in which students process examples has a strong effect on their later problem solving (Chi, Bassok, Lewis, Reimann, & Glaser, 1989; VanLehn, Jones, & Chi, 1992). In a recent series of experiments, we examined the ways in which processing examples can be useful in acquiring problem solving knowledge (Trafton, 1994; Trafton & Reiser, 1993). We argued that students need to apply the knowledge gathered from studying examples to solve later problems in order to form useful problem solving skills (cf. Anderson, 1987). The effectiveness of studying annotated examples was reduced when a related problem to solve did not immediately follow the studied example, suggesting that drawing upon information studied from an example to construct one's own solution to a new problem is an important component of acquiring problem solving knowledge.

These results emphasize the importance of the manner in which students process instructional examples. An important approach in this work has been to manipulate students' study activities on a set of *source* problems, and then examine the effects of working with those sources on their ability to solve later analogous *target* problems. Source and target problems overlap in the knowledge and subskills needed to solve them. What influences the effectiveness of a source problem in guiding later problem solving? Sweller and Cooper (1985) argued that studying worked examples as sources is more effective than solving the same problems. They found that studying examples interleaved with problem solving drawing upon those sources produced more effective performance on later posttests than solving those same sources. In contrast, Trafton (1994) found that solving sources led to superior posttest performance, and produced faster problem solving on the targets than studying the same example sources.

Clearly, there are many factors that might influence the relative effectiveness of encoding source problems through study or solving them oneself. Trafton and Reiser (1993) argued that there is a tradeoff between the active nature of constructing solutions, in which students must perform subgoal decomposition, operator selection, and execution, and the facilitating effect of guiding problem solving with a worked example. If the search space is extensive, then the guidance of an example may greatly facilitate problem solving, and it may be difficult to map from a solution constructed with much patching and debugging. Yet, if the problem solving can be made more productive by reducing the overhead of constructing solutions, making them more easily interpretable, then the additional practice of generating solutions may be more profitable. Trafton (1994) found that when the learning environment provided sufficient support for instrumental parts of the task (e.g., minimizing the cognitive load of syntax and debugging in programming), the extra practice involved in constructing solutions to source problems outweighed the benefits of the guidance provided by worked examples.

The strategies used to process examples may also mediate their effectiveness relative to solving problems. Subjects who take a more active role in studying solutions, attempting to explain each component, considering why it was selected and how it operates, learn more from studying those examples (Chi, et al., 1989; VanLehn, et al., 1992). Thus, the degree to which subjects treat examples as problems to be

mentally solved affects how well those examples can function in guiding later problem solving. The initial evidence for the effectiveness of self-explanation has been correlational, relying on classifying more and less successful problem solvers and looking for associated differences in their self-explanation behavior on examples in the lesson (Chi, et al., 1989; Pirolli & Recker, 1994). Recently, there has also been an attempt to train self-explanation and see whether subjects thus trained perform better in later lessons (Bielaczyc, Pirolli, & Brown, 1994).

How does self-explanation affect the relative efficacy of studying examples versus solving problems? Again, we focus on the use of knowledge acquired from processing a source problem to facilitate constructing solutions to target problems. Can self-explained examples as sources function as effectively as solving those problems themselves? The present experiment examines this question, using a supportive learning environment for LISP programming, in which we had previously observed superior performance with solved sources (Trafton, 1994). Second, we consider whether self-explanation also affects the efficacy of a solved problem. We are exploring the extent to which self-explanation can be supported in the structure of the learning environment activities. In our work on learning environments, we have argued that environments can be constructed to be congruent with effective reasoning strategies (Merrill & Reiser, 1994). In the present experiment, we examine the support a learning environment can provide for self-explanation strategies.

Method

Design

We examined novices learning to program in LISP. We presented subjects with pairs of source and related, but not isomorphic, target problems which overlapped in the subskills necessary for their solution. We manipulated the type of source problem (worked example or problem to solve) and whether subjects were given instructions to self-explain while studying or solving the source problems. Source and target problems were interleaved to enable examination of the effects of the source study method on solving problems requiring overlapping subskills. A sample source/target pair is shown below:

- *Source Problem:* A sales company has planned all the routes of its sales personnel. Then management decided that all sales personnel must begin each trip at the nearest branch office instead of whatever they had originally planned as a first destination. Write a function that takes two inputs -- the new destination, e.g. *chicago*, and a list of old destinations, e.g. *(detroit cinci stl)*, and replaces the first destination of the list with the new destination. The output should be the new route -- e.g., *(chicago cinci stl)*.
- *Target Problem:* While a university admissions worker was entering personal information about students, it was discovered that many last names were entered incorrectly. To simplify future data input, write a function that takes a correct last name, e.g., *smith*, and a list of personal information, e.g., *(john smit 24 psychology)*, and returns a new, corrected list with the last name replacing both the first and last name of the old list, e.g. *(smith 24 psychology)*. Your function should take two inputs: the first should be the correct last name (an atom), and the second should be the old information (a list).

Subjects in the Example conditions studied a problem statement, presented with a motivating cover story, and a solution presented with no annotation other than a sample input and its output. These subjects also constructed the example solutions in the editor, to equate for interface practice in building, editing, and running programs across all conditions. Subjects in the Solve conditions saw the same source problems but solved them entirely on their own. All conditions solved the same target problems, which were indistinguishable in presentation from the source problems of the Solve conditions. All problems were taken from Trafton (1994), and were modified to exclude the annotations used in this earlier study; piloting showed that subjects were at a loss as to what to self-explain when given fully annotated examples. All problems were presented in the same order to all subjects. The design is summarized in Table 1.

Table 1. Design of the learning sessions

	No Self-Explain	Self-Explain
Example	Studied example sources; solved targets	Explained aloud example sources; solved targets
Solve	Solved sources and targets	Solved and explained aloud sources; solved targets

Apparatus and Materials

Subjects worked with VSE, an interactive learning environment for LISP which provides significant support for the operational aspects of programming: ensuring legal syntax and providing strong support for testing and debugging programs. (VSE is described more completely in Merrill & Reiser, 1994). Programs are built in VSE by dragging functions from a menu and placing them into an initially skeletal function body (containing a *defun* form with a given name and empty parameter list), significantly simplifying the code construction process. VSE also enables students to run their programs on test inputs, either all at once or step-by-step. Solutions are then submitted to VSE, which

informs students if their solutions are correct or not. Further, a debugging probe allows students to inspect intermediate function output values during a run. Finally, VSE provides simple hints on legal use of functions when errors arise while running programs. VSE allows students to focus less on the operational aspects of programming LISP functions, and more on the semantics of combining functions. For the Self-Explain conditions, VSE was slightly modified to prompt subjects to predict the output of their functions, based on the input values they entered, immediately prior to any code being evaluated. In this way, the justifications of their code elicited through self-explanation were immediately tied to their expectations about program behavior. This should enhance any effects of self-explanation by instantiating any newly generated domain knowledge in explicitly visible program behavior.

Procedure

Subjects participated in a single session which they began by reading the first chapter from an introductory LISP textbook (Anderson, Corbett, & Reiser, 1987). The text described some basic LISP functions, the role of functions in programming, and the use of variables. Subjects retained this text for reference throughout the acquisition phase of the experiment. Following the reading, subjects in the Self-Explain conditions were given a brief explanation of the purpose of self-explanation and some initial practice in self-explanation techniques. To ground this explanation, these subjects solved a constraints satisfaction problem (from Nathan, Mertz, & Ryan, 1994), unrelated to LISP or programming. Subjects in the Example Self-Explain condition were given this practice problem and a step-by-step solution which they were asked to explain. Subjects in the Solve Self-Explain condition were given the same problem without a solution, and asked to solve it and justify each step in their solution.

Subjects in the two Self-Explain conditions were then provided instruction on the purpose of self-explanation and the kinds of things to explain when solving a LISP problem. Specifically, subjects were instructed to explain the purpose of each function used in a solution with respect to the requirements stated in the problem, and how that function achieved its purpose. Subjects were also instructed to explain the role of any variables used in the problem, including what information that variable held upon entry to the function and how it was operated on. At this time, subjects were given a list of strategic questions from Bielaczyc et. al (1994) and instructed that asking themselves these questions would help them to construct self-explanations. This explicit instruction in self-explanation was intended to control the kind and quality of elaborations subjects produced. In particular, we aimed to maximize the quality of self-explanations in order to maximize its benefits.

Subjects in all four conditions then received a demonstration of the computer system, working through two problems (one source, one target) with the experimenter. Subjects then worked through five source-target problem pairs, uninformed of the source vs. target distinction. All subjects worked on each problem until correct. In the Self-Explain conditions, subjects read each source problem aloud and then proceeded to explain either the solution given (Example) or their own solution (Solve). These source problems were solved in front of the experimenter, who prompted subjects only when necessary to clarify vague statements, when subjects paused talking for more than a few seconds, or when subjects needed to speak more loudly. Subjects were not required to explain solutions in any particular order. For the Example Self-Explain condition, this meant that subjects could explain the role of any function in any order, not necessarily from left to right. For the Solve Self-Explain condition, subjects were encouraged to explain each step as they went. Subjects were required, however, to explain all portions of their code they had not yet explained prior to their first run of a new or modified solution. Following the successful solution of the source, subjects were left alone to solve the next target problem.

During the acquisition phase, subjects received help from the experimenter only for questions related to the operation of VSE. Subjects were referred to the text for LISP-related questions. Direct help on any problem was provided only if subjects took an inordinate amount of time to reach solution (more than 20 minutes on a single problem). Help was always given as hints, never as direct suggestions as to what to do next, or which function to use, etc. Help was offered by the experimenter only after a request from a subject, but was actually rarely sought.

Posttest: Following the acquisition phase, all subjects completed the same posttest. Subjects were not instructed to self-explain during the posttest, but they were also not discouraged from doing so. During the posttest, the reference text read prior to the acquisition phase was unavailable, as was the VSE feature allowing functions to be run and debugged. Subjects constructed what they thought was the correct solution to a problem, and submitted it without feedback on its correctness from the system or experimenter.

Subjects

Subjects were 40 Northwestern University undergraduates with one quarter course or less of computer programming experience. No subject had prior experience with LISP. Data from one potential subject were not included in these analyses because the subject exhibited substantially more difficulty with the material than other subjects (as indicated by solution time and errors), and appeared to be engaging in a guessing strategy (submitting untested answers) different from other subjects. Of the 40 subjects, 26 were female and

14 were male. The age of the subjects ranged from 18 to 22 years, mean 20 years.

Subjects were randomly assigned to condition so as to approximately balance mathematics SAT scores. The mean math SAT scores of each condition were: Example No-Self-Explain 673, Example Self-Explain 675, Solve No-Self-Explain 676, and Solve Self-Explain 677. The median math SAT score for all conditions was 670.

Results and Discussion

We were interested in two aspects of subject performance: difficulty in solving target problems and posttest performance. First, we considered the time subjects took to solve the target problems as a measure of the relative utility of their work on the preceding source problem. We performed a 2x2 analysis of covariance of source problem type (example or solve) and source instruction (self-explain or not), using math SAT score as the covariant. There were no main effects of either source problem type or source instructions on target problem solution time. The pattern of results, shown in Figure 1, suggests differential effects of self-explanation depending on source instruction, but the interaction was only marginally reliable, $F(1, 35) = 2.42$, $p = .13$. Because we expected there may be differential effects of self explanation depending upon whether subjects studied or solved the source problems, we computed two planned comparisons to separately examine the effects of self-explanation for each of the Example and Solve conditions. Also, to examine more closely exactly where subjects' time was being spent, we broke target solution time down into four components: initial planning time (time spent looking at a problem prior to any activity), build time (the time spent inserting new functions or variables into code), edit time (deleting and replacing code), and testing time (time spent running programs).

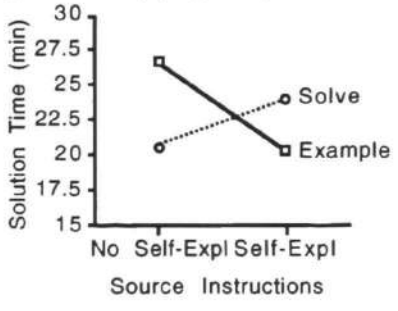

Figure 1: Time to Solve Targets

Of the four components of target solution time, the interaction of source instruction and problem type was reliable for editing time, $F(1, 35) = 4.9$, $p < .05$ (Figure 2), and not for the other three components. The planned comparison revealed that subjects who self-explained examples spent significantly less time editing their solutions, $F(1,35) = 3.8$, $p = .05$. In contrast, there were no effects of self-explanation for subjects solving source problems, $F<1$ on any target solution time components.

Subjects who self-explained examples also displayed fewer errors during program runs, 1.3 vs. 3 errors, although this trend was not reliable, $F(1,35) = 1.96$, $p = .17$. Consistent with this, Example subjects deleted somewhat fewer program components on targets if they self-explained the sources, 6.3 vs. 13.1 deletes, although this trend also was not reliable, $F(1,35) = 1.56$, $p > .20$. These results suggest that subjects who did not self-explain examples may have performed more edits or spent more time deciding how to repair their programs than subjects that did self-explain.

Taken together, the time and error trends suggest that subjects who self-explained examples were able to more effectively encode the relevant knowledge and subskills from source problems and apply them successfully to subsequent target problems with fewer errors. Indeed, as Figure 1 suggests, this improved encoding improved the performance of the Example Self-Explain condition to the level of the Solve conditions. Interestingly, there was no evidence that self-explanation benefited the subjects who solved the source problems.

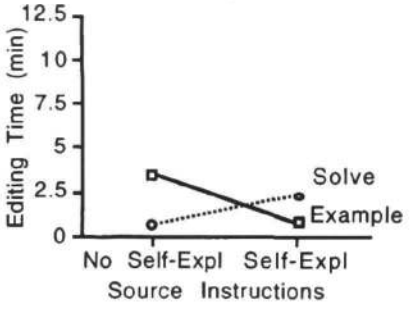

Figure 2: Time Editing Targets

We also examined the posttests to determine the effects of source study method on later performance. There were no main effects of source problem type or source instruction or an interaction on posttest score, all $F<1$ (see Table 2). It seems likely that our subjects reached a ceiling effect on posttest score, since all subjects eventually correctly solved all the problems in the acquisition phase, using the debugging tools available in the environment. There was, however, a significant interaction of source type and instruction on the time to construct solutions on the posttest, $F(1,35) = 5.38$, $p < .05$. Again, this interaction was due to the improved performance of the Example subjects who self-explained sources over those who did not, $F(1,35) = 8.39$, $p < .01$, while there was no effect of self-explanation on the Solve conditions, $F<1$. Thus, the benefit of self-explaining examples carried over to problem solving efficiency on the posttest. Self-explanation while studying examples improved performance, as evidenced by faster solutions, and improved it to the level of subjects who solved the original sources.

Table 2: Performance on the Posttest

	Example No Self-Expl	Example Self-Expl
Score	78.0	80.4
Time (min)	19.0	13.6
Deletes	9.0	5.7
	Solve No Self-Expl	**Solve Self-Expl**
Score	83.0	81.4
Time (min)	15.7	16.2
Deletes	5.1	6.6

How did the self-explanations while studying examples led to greater problem solving efficacy? VanLehn et al. (1992) proposed that the elaborations generated during self-explanation provide new knowledge which can be used to guide later problem solving. Our findings suggest this is indeed the case within this environment. Recall that subjects were explicitly instructed to provide domain-based justifications for the use of functions: why one would want to use a particular function in a certain place, and how that function would achieve a subgoal of the problem. These subjects' self-explanation activity was thus focused on encoding both the operational behaviors of LISP functions and the kinds of subgoals that pieces of code could accomplish. This encoding then enabled them to better map the knowledge acquired studying examples to analogous situations in target problems. Thus, self-explanation of examples allowed subjects to more accurately guide their search of the solution space during target problems, as reflected by the lesser amount of time spent editing erroneous solutions. Furthermore, subjects not only elaborated their understanding of why a particular example solution would work, but also tested their predictions of the outcome of each solution component. Thus, subjects could test and debug these encodings prior to subsequent problem solving. We believe that this explicit support within the learning environment for testing predictions contributed to the efficacy of self-explanation for these subjects.

The results in Figure 1 are consistent with the findings of Trafton (1994), in which students solving source problems outperformed students studying examples. Here, self-explanation of examples seems to have improved performance to the level exhibited by the Solve conditions in these studies. Subjects' self-explanation of examples allowed them to construct the kind of problem solving knowledge that the Solve condition subjects were generating by constructing their own solutions. Consistent with our earlier arguments (Trafton, 1994; Trafton & Reiser, 1993), subjects who self-explained examples were actively constructing solutions similar to Solve subjects, while benefiting from the guidance of a completed solution they were attempting to explain that they knew to be correct. Being high on both levels of this tradeoff appears to be an advantage for subsequent problem solving.

We must ask, of course, why self-explanation had no positive effect for subjects solving source problems. The explanation may be simply that, within this environment at least, the act of generating solutions to problems led to the acquisition of subsequently relevant problem solving knowledge, and there was little room for self-explanation to have any benefit. VSE provides significant support for problem solving, effectively reducing the search space subjects must traverse to build solutions. In such a constrained problem space, the need for strategic monitoring of problem solving is reduced, thus reducing a major benefit of self-explanation. Furthermore, the domain problem solving knowledge that subjects self-explaining examples were constructing was already being directly generated by subjects in the Solve conditions. Thus, the elaborations elicited by self-explanation may have been no more productive than the elaborations subjects were already constructing to select and execute plans and operators.

The results here are consistent with the results from other recent studies demonstrating an advantage for self-explanation (Bielaczyc, et al., 1994; Nathan, et al., 1994; Pirolli & Recker, 1994). Both Bielaczyc et al. and Pirolli and Recker measured performance gains as we did on target problems: the number of errors on subsequent problem solving trials, rather than on posttests. We do not see the improvement in posttest scores found by Nathan, although it seems most likely that our subjects reached a ceiling effect on the posttest due to the substantial feedback provided by VSE and the requirement that subjects correctly solve all problems in the learning session.

In summary, our results show a marked trend toward improved performance for those subjects who self-explain examples over subjects employing their default study strategies. One may wonder why subjects' decreased editing time is not significantly reflected in our other measures of the problem solving process, such as the time to plan, build, or test solutions. One possible explanation is that the intervention described here is quite short. Subjects had minimal instruction on self-explanation and were able to practice it on only five problems. In Bielaczyc et. al's (1994) study, for example, subjects spent over 12 hours problem solving, with more than an hour devoted solely to learning self-explanation strategies.

Still, even with the minimal training provided here, subjects who self-explained worked examples were able to more efficiently solve target problems and posttest problems. Our results suggest that the benefit these subjects gained was a clearer understanding of what each function they had learned did and how they could be combined. That is, subjects who self-explained examples built correct target solutions with less effort spent repairing erroneous solutions than their counterparts who merely studied examples. The

elaborations that these subjects produced enabled them to construct relevant problem solving knowledge as if they had solved the problems themselves.

The benefit of self-explanation appeared only when subjects had worked examples to explain. Subjects in the Solve conditions had to do considerably more work to generate source solutions than their example counterparts. This work apparently paid off during target problem solution, regardless of whether it was self-explained. Self-explanation did not appear to affect subjects' ability to generate solutions on their own.

Conclusion

The present study complements our earlier studies (Faries & Reiser, 1995; Trafton, 1994; Trafton & Reiser, 1993) showing that the effectiveness of the study of examples as a method of skill acquisition is critically related to students' ability to apply the knowledge gained through such study to problem solving practice. We have previously argued that studying examples is not in itself enough to ensure useful learning in such domains as programming; examples must be actively and productively applied to new problems to be effective. This study shows that self-explanation enhances the utility of examples and students' ability to more efficiently apply new knowledge gained through such study of worked examples to new problems.

It is unclear from our current findings whether self-explanation can, under some conditions, have a beneficial effect for students trying to solve problems on their own. Perhaps the strategies for self-explaining self-generated solutions are different than for self-explaining worked examples. If such is the case, it has important implications for inquiry learning, where students are often generating not only their own solutions, but their own problems. Here, the ability to successfully monitor one's own problem solving activity becomes paramount. We are investigating these issues in a considerably less procedural domain: scientific reasoning (Tabak, Sandoval, Smith, Agganis, Baumgartner, & Reiser, 1995). We expect that supporting students in articulating and explaining their own inquiry will be crucial to their success.

Acknowledgments

The learning environment reported in this paper was constructed by Adnan Hamid and Douglas Merrill. This research was supported in part by grants N00014-91-J-1125 to Princeton University and N00014-93-1-0136 to Northwestern University from the Office of Naval Research. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of these institutions. We thank Susan Williams for helpful comments on earlier drafts. See URL http://www.ls.sesp.nwu.edu/Learning_Sciences/faculty/reiser.html for related work.

References

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*(2), 192-210.

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

Bielaczyc, K., Pirolli, P., & Brown, A. L. (1994). *Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem-solving* (Report No. CSM-7). University of California at Berkeley.

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.

Faries, J. M., & Reiser, B. J. (1995). The encoding and retrieval of problem solving episodes. In preparation, The Institute for the Learning Sciences, Northwestern University:

Merrill, D. C., & Reiser, B. J. (1994). Scaffolding effective problem solving strategies in interactive learning environments. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, (pp. 629-634). Atlanta, GA: Erlbaum.

Nathan, M. J., Mertz, K., & Ryan, R. (1994). *Learning through self-explanation of mathematics examples: Effects of cognitive load.* Paper presented at AERA, New Orleans, LA.

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

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*, 257-285.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*, 58-89.

Tabak, I., Sandoval, W. A., Smith, B. K., Agganis, A., Baumgartner, E., & Reiser, B. J. (1995). Supporting collaborative guided inquiry in a learning environment for biology. In review:

Trafton, J. G. (1994). *The contributions of studying examples and solving problems to skill acquisition.* Ph.D. Thesis, Princeton University.

Trafton, J. G., & Reiser, B. J. (1993). The contributions of studying examples and solving problems to skill acquisition. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, (pp. 1017-1022). Boulder, CO:

VanLehn, K., Jones, R., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*, 1-59.