Learning Computer Programming: The Effects of Collaborative Explanations and Metacognition on Skill Acquisition

Katerine Bielaczyc
Education in Math, Science and Technology
University of California at Berkeley
kateb@dewey.soe.berkeley.edu

I. OBJECTIVES

An important aspect of learning in problem-solving domains is how students initially acquire the concepts and procedures from instruction and apply the knowledge to solving associated problems. If we look at individual learners, we find that some are more successful than others in learning and problem solving. Empirical studies in the domains of computer programming (Bielaczyc, Pirolli & Brown, 1993; Pirolli & Bielaczyc, 1989) and physics (Chi, Bassock, Lewis, Reimann, & Glaser, 1989; Fergusson-Hessler & deJong, 1990) have found that particular types of learning activities during initial knowledge acquisition are related to performance on subsequent problem solving tasks. These investigations focussed on the activities of students working individually to study instructional materials prior to problem solving. The present study broadens this scope to investigate the activities of students working collaboratively to study. The research provides a fine-grained examination of collaborative explanations and metacognition in the course of learning from expository texts and examples and identifies specific activities that relate to successful learning.

II. RESEARCH FRAMEWORK

The underlying framework for the research is aimed at integrating models of active, goal-oriented learning processes with theories of problem-solving and cognitive development (e.g. Anderson, 1987; Bereiter & Scardamalia, 1989; Brown & Palincsar, 1989). The educational context involves studying instructional texts and examples and solving associated problems. The genesis of this line of research was a study on the self-explanations, or student-generated elaborations, made by learners working individually to study materials on LISP programming. This research (Pirolli & Bielaczyc, 1989) found that particular quantitative and qualitative characteristics of the self-explanations made while studying instructional materials correlated with students' subsequent problem-solving performance. A more recent study (Bielaczyc, Pirolli, & Brown, 1993), strengthened these correlational findings by showing experimentally that training students in the types of self-explanation and self-regulation strategies found to be used by high-performers resulted in improvements in learning and problem solving performance. The present research applies the earlier theoretical and experimental framework to the study of collaborative explanations and metacognition.

III. EXPERIMENTAL DESIGN

The context for the studies was a self-paced programming course on LISP programming. The CMU LISP Tutor (Reiser, Anderson, & Farrell, 1985) was used to collect fine-grained problem solving measures. Although the use of an intelligent tutoring system is not common, the general lesson structure is typical: studying instructional materials involving expository texts and examples then solving associated programming exercises.

University students or recent graduates with no prior programming experience participated in the study (N = 25). There were three main phases to the study: (1) a set of three introductory LISP

programming lessons, (2) a Pre-Target lesson on helping functions, and (3) a Target lesson on recursion. Each subject typically spent 15 hours learning programming. The experimental sessions were videotaped, and the subjects were requested to "think aloud" while reading instructional materials and solving programming problems.

All subjects worked through the introductory and Pre-Target lessons individually. Subjects were divided into groups of high-performers (H) and low-performers (L) based on their performance on the Pre-Target lesson. Orthogonal to this grouping, subjects were divided into groups that either studied collaboratively in HL dyads (dyad group) or studied individually (individual group) for the Target lesson. Following studying the instructional materials, all subjects worked individually on the Target lesson programming exercises.

V. ANALYSES AND RESULTS

The investigation included both quantitative and qualitative analyses of the programming performance measures and the video protocols collected as subjects proceeded through the phases of the instruction. The present focus is on the results of a set of within-group analyses of the collaborative dyad group. Of interest is whether the performance differences between collaborative dyad subjects for the Target lesson exercises are related to the types of explanations subjects generated while studying the Target lesson instructional materials.

The collaborative dyad group was divided into equal-sized subgroups of *high-benefit* and *low-benefit* dyad subjects based on Target lesson programming performance. A set of comparative analyses between high-benefit and low-benefit dyad groups was then performed based on the features of the explanations generated by dyad subjects while studying the Target lesson instructional materials. The analyses focussed on:

(a) the strategic aspects of the explanations,
(b) the content of the explanations, and
(c) the manner in which explanations were generated.

High-benefits in performance were found to correlate with particular explanation and monitoring strategies. Specifically, high-benefit dyad subjects show more evidence of elaborating the main ideas introduced in the textual portions of the manual, making more connections between abstract ideas presented in the text and their instantiations in the example, and focussing on the recursion-related aspects of the example code. These results suggest that high-benefit subjects are producing elaborated representations of the recursive concepts and are building up an understanding of the meaningful relations underlying the examples. High-benefits in performance were also found to correlate with higher amounts of elaborations and inferences focussing on issues central to understanding recursive functions: (a) structural issues, (b) operational issues, and (c) coding issues. This suggests that high-benefit subjects are producing more information relevant to subsequent programming tasks.

Using one's partner as a question-asking resource during comprehension failure appears to provide benefits to both members of the collaborative dyad. The subjects that frequently asked for and received explanations from their partners and those that provided explanations to their partners' questions were found to have high-benefits in subsequent performance. These results are consistent with the findings in the literature (e.g. Webb, 1985, Peterson & Swing, 1985). By asking direct questions and receiving an explanation, question-askers may be able to overcome points of confusion and comprehension failure and acquire information that may serve to fill in gaps in understanding. Conversely, by providing explanations, question-answerers may themselves acquire a deeper understanding of the materials. However, a close examination of the question-asking episodes suggests that a question-asker benefits most from a response when (a) the question-asker subsequently incorporates the received information into self-generated domain elaborations, and (b) the response is at the appropriate level for the question-asker.

VI. CONCLUSION

In learning computer programming, the types of activities in which students engage in the course of learning from instructional materials play an important role in their ability to write their own computer programs. The present study contributes to our understanding of the types of collaborative explanation activities that students engage in while studying instructional texts and examples and identifies specific activities that relate to successful problem solving. The results highlight the complex interactions among a learner's knowledge, learning strategies, and the resources provided by a co-collaborator. The characterisation and articulation of successful learning activities are important, as ultimately we'd like to be able to address issues such as:

- How can we help students to engage in effective learning activities?
- How should we design collaborative technologies?

VII. REFERENCES

Bielaczyc, K., Pirolli, P., & Brown, A.L. (1993) Strategy training in self-explanation and self-regulation strategies for learning computer programming, Technical Report CSM-5, UC Berkeley.

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

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.

Peterson, P.L., & Swing, S.R. (1985) Student's cognitions as mediators of the effectiveness of small-group learning. *Journal of Educational Psychology*, 77 (3), 299-312.

Pirolli, P. and Bielaczyc, K. (1989), Empirical analyses of self-explanation and transfer in learning to program. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 450-457, Hillsdale, NJ: Erlbaum.

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

Scardamalia, M., and Bereiter, C. (1991). Higher levels of agency for children in knowledge building: A challenge for the design of new knowledge media, *The Journal of the Learning Sciences*, 1 (1), 37-68.

Webb, N.M. (1985) Student interaction and learning in small groups: A research summary. In R. Slavin, S. Sharan, S. Kagan, R. Hertz-Lazarowitz, C. Webb, and R. Schmuck (eds) *Learning to Cooperate, Cooperating to Learn* , New York: Plenum Press, 147-172.