Investigating Novice Programmers' Plan Composition Strategies

Francisco Enrique Vicente G. Castro
Worcester Polytechnic Institute
Department of Computer Science
Worcester, Massachusetts, 01609, USA
fgcastro@wpi.edu

ABSTRACT

Problem solving through effective plan decomposition and composition continues to be exceedingly difficult for novice programmers. This is exacerbated by the fact that these strategies are usually implicit in instruction: students are left to figure out their own problem solving strategies. My research investigates ways to elicit and improve students' plan decomposition and composition strategies.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education—*Computer Science Education*.

Keywords

Novice programmers, problem solving, plan composition

1. PROGRAM CONTEXT

I am beginning the second year of my PhD program in Computer Science at WPI. I am a full-time PhD student and research assistant under the supervision of Dr. Kathi Fisler. I anticipate completing my research qualifying milestone by Fall 2015 and my PhD breadth requirement by Spring 2016 (to gain candidacy). To inform my research, I have included learning science courses in my program coursework. I completed my master's degree in 2014.

As part of my research qualifier, I have completed an initial study of plan composition behavior of CS1 students at WPI. The study and preliminary findings will contribute to the development of a full dissertation proposal around this topic in the coming academic year.

2. CONTEXT AND MOTIVATION

A problem being explored in computing education is plan composition: how students analyze a problem, decompose it into finite subproblems, and finally weave solutions to these subproblems into a coherent, working program. Most introductory courses emphasize using programming concepts in code, usually through extensive sets of exercises [1, 4]. Students are then expected to develop general problem solving strategies, yet they often do not know where or how to start [8]. They are unable to identify problem components and relevant algorithmic structures, as well as the interactions between these components [1, 4, 5, 6].

These difficulties are often not due to students' lack of understanding of how to use language constructs. The problem stems from the fact that problem solving and plan composition strategies are usually implicit in instruction; students are left to figure out their own plan composition strategies [1, 4]. Soloway's study on the Rainfall problem [8] highlights the exceeding difficulty students experience in carrying out plan composition effectively; the problems that underlie plan composition [9] continue to be evident even now.

3. BACKGROUND & RELATED WORK

Literature highlights that novices not only need to be taught programming language syntax and semantics, but also need explicit instruction on design skills, problem solving, and plan composition [8, 10]. Spohrer and Soloway identify both novices' language construct-based difficulties and an extensive list of problems that make plan composition complicated for novices [9]. Their findings suggest that the occurrence of bugs are due to students' inability to effectively organize and integrate the goals and plans that underlie program code, a sentiment echoed by Pirolli et. al. in their study on learning recursive programming [7]. Both of these projects assume that people approach new problems by recalling and modifying solutions to similar problems [7, 10]. These and later projects try to give plans explicit labels to help students recognize (a) concrete goals of individual subproblems and (b) patterns across similar problems [6, 8].

How to Design Programs (HTDP) [2] is an introductory computing curriculum that approaches similarity differently from the projects discussed previously. HTDP teaches students to produce a common code shape based on the structure of the input data, then to use examples to tailor the common shape to a specific problem. Fisler [3] investigated plan composition behavior in HTDP students. These students made fewer errors than in earlier Rainfall studies, but several still struggled with plan composition. My work will explore how the combination of HTDP's data-driven methodology and others' labeling oriented approaches might be used together to improve students' plan composition skills.

ICER'15, August 9–13, 2015, Omaha, Nebraska, USA.
ACM 978-1-4503-3630-7/15/08.
http://dx.doi.org/10.1145/2787622.2787735 .

4. STATEMENT OF THESIS/PROBLEM

The context and and related work informs and guides my research question: When and how do students' task decomposition and code composition instances emerge in their programming process and how can we influence them to do it better? Interesting sub-questions include:

1. Do students: *a*) decompose a problem into smaller plans, implement the individual tasks, and then recompose; *b*) implement a task-related piece of code and then patch around it; or *c*) pull out related code (possibly from prior work done), modify the code to adapt to the problem context, and then build around it?
2. What role can design-based approaches like HTDP play in how students implement task decomposition and code composition strategies?

5. RESEARCH GOALS & METHODS

Methods I expect to use include:

Elicit students' problem solving strategies. We're interested both in the code students produce and more talk-aloud style data. Students' programming activity will be video captured for analysis. I will analyze these for instances of plan decomposition, recomposition, and other relevant problem-solving behavior. I will also use a combination of surveys or other free-form input to understand how students are approaching these problems. My initial study used a combination of these methods.

Elicit program design principles students use in plan composition. How students think about composition can be reflected both in the code they write and on how they comprehend solutions others have written. Data on a combination of problems on writing code and comprehending code will be collected from several educational institutions to understand students' perceptions of composition.

Investigate the role of cognitive factors in plan composition strategies. Because learning programming imposes cognitive load upon the learner, further review of literature will be done to consider how cognitive factors play a role in problem solving.

6. DISSERTATION STATUS

For my research qualifier, I have completed an exploratory study of the edits students made while working on a plan composition problem. We video captured the programming activity of CS1 students at WPI. Preliminary results suggest that while students' programming is informed by HTDP and students recognize the need for plan decomposition, they struggle with the process of how to decompose the problem into coherent plans and eventually recompose solutions. In the cohort that was analyzed, plan decomposition was very minimal, tests were used to confirm code outputs instead of informing the design of solutions, and recomposition could not be effectively carried out due to the ineffective ways with which plans were developed in the first place. The study involved the development of a coding scheme which will be modified as the research progresses.

From the doctoral consortium, I hope to gain insight from commentaries of my research topic and the initial study I have conducted, as well as suggestions to methodology and additional ideas and perspectives of the subject matter that I may have not examined so far. Likewise, feedback may help refine my research questions. Findings from future data analyses using new data to be collected from other institutions will be used for the dissertation proposal which I hope to submit and defend within the year.

7. EXPECTED CONTRIBUTIONS

The contributions of this work to the computing education community include the identification of students' plan decomposition and composition methodologies and techniques that might influence these methodologies. The findings from this research may further inform the development of future computing education curricula and pedagogy.

8. REFERENCES

[1] M. de Raadt, M. Toleman, and R. Watson. Training Strategic Problem Solvers. *SIGCSE Bull.*, 36(2):48–51, June 2004.

[2] M. Felleisen, R. B. Findler, M. Flatt, and S. Krishnamurthi. *How to Design Programs: An Introduction to Programming and Computing*. The MIT Press, Cambridge, Mass, Feb. 2001.

[3] K. Fisler. The Recurring Rainfall Problem. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 35–42, New York, NY, USA, 2014. ACM.

[4] A. Keen and K. Mammen. Program Decomposition and Complexity in CS1. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 48–53, New York, NY, USA, 2015. ACM.

[5] J. Mead, S. Gray, J. Hamer, R. James, J. Sorva, C. S. Clair, and L. Thomas. A Cognitive Approach to Identifying Measurable Milestones for Programming Skill Acquisition. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '06, pages 182–194, New York, NY, USA, 2006. ACM.

[6] O. Muller, D. Ginat, and B. Haberman. Pattern-oriented Instruction and Its Influence on Problem Decomposition and Solution Construction. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '07, pages 151–155, New York, NY, USA, 2007. ACM.

[7] P. L. Pirolli, J. R. Anderson, and R. G. Farrell. Learning to program recursion. In *Proceedings of the Sixth Annual Cognitive Science Meetings*, pages 277–280, 1984.

[8] E. Soloway. Learning to Program = Learning to Construct Mechanisms and Explanations. *Commun. ACM*, 29(9):850–858, Sept. 1986.

[9] J. C. Spohrer and E. Soloway. Novice Mistakes: Are the Folk Wisdoms Correct? *Communications of the ACM*, 29(7):624–632, July 1986.

[10] J. C. Spohrer and E. Soloway. Simulating Student Programmers. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pages 543–549, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.