Exploring Student Progressions in Multiple, Interrelated Program Design Skills

Francisco Castro[1] and Kathi Fisler[2]
Worcester Polytechnic Institute[1], Brown University[2]
fgcastro@cs.wpi.edu, kfisler@cs.brown.edu

Abstract:

Learning effective program design remains a nontrivial goal for novice programmers in introductory computing courses [1, 4]. While different courses may use different programming languages and problem domains, the underlying program-design skills are fairly common, and include selecting appropriate language constructs, composing code fragments for multiple problem tasks, and checking whether the resulting program satisfies the original problem constraints. How to Design Programs (HTDP) [2] is an introductory computing curriculum that has been adopted in higher education institutions and some K-12 programs. While it fosters similar skills as other curricula, HTDP uses a unique pedagogy for teaching these skills using a multi-step process of program design; how students learn with HTDP remains largely unexplored in CSEd research.

We report on the first phase of a project that explores the evolution of students' design skills in a two-course introductory sequence, as framed by their learning context (HTDP). We interviewed and conducted think-alouds with students at certain points during the course of their CS1 and through to CS2. An analysis of the CS1 data yielded a multi-strand SOLO-based framework [3] for multiple, interrelated program design skills and the progressions of these skills, alongside several factors that are not amenable to a SOLO progression.

References:

[1] Francisco Castro and Kathi Fisler. 2016. On the Interplay Between Bottom-Up and Datatype-Driven Program Design. In *Proceedings of SIGCSE '16*. New York, NY, USA: ACM, 205–210.

[2] Matthias Felleisen, Robert Findler, Matthew Flatt, and Shriram Krishnamurthi. How to Design Programs. MIT Press, 2001. http://www.htdp.org/.

[3] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline Whalley. 2008. Going SOLO to Assess Novice Programmers. *In Proceedings of ITiCSE '08*. New York, NY, USA: ACM, 209–213.

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