Identifying and Assessing Computational Thinking Practices

Wendy Martin, Education Development Center, Inc., wmartin@edc.org
Karen Brennan, Harvard University, karen_brennan@gse.harvard.edu
William Tally, Education Development Center, Inc., btally@edc.org
Francisco Cervantes, Education Development Center, Inc., fcervantes@edc.org

Abstract: Informal programs that allow students to be digital creators appeal to educators who want students to experiment, innovate, and solve problems. To integrate such experiences into formal contexts, educators must name, capture, and assess what students learn from creating in ways that schools value. This poster presents resources to help teachers understand what computational thinking practices look like, how to collect student work that demonstrates those practices, and how to assess that kind of work.

Major Issue Addressed

Some of the most innovative educational programs are in informal settings—afterschool programs, museums, and youth organizations—where young people are encouraged to explore, discover, and create without the time limitations and curricular and assessment requirements that exist in formal education environments. Freedom to experiment is particularly important for young people learning to create computational media. Scratch is a graphical programming language that has been used in informal settings for a number of years, enabling young people to engage in the process of designing and programming their own projects and sharing them with friends, family, and the online Scratch community. Over the past few years there has been mounting interest in the K-12 education community in helping students develop the habits of mind described as computational thinking (CT), such as systematic problem solving, iterative design, and abstracting from patterns (National Research Council, 2011, 2012). Some teachers and schools have opted to integrate Scratch into their instruction to support the development of CT. However, there are significant challenges to adapting programs originally used in informal contexts to formal schooling on a broader scale (NRC, 2011). One particular obstacle is that it is difficult for school districts to commit to integrating content that they cannot readily assess (Grover & Pea, 2013), and currently there are no comprehensive assessments of CT applicable for a wide range of K-12 educational contexts. This poster is intended to initiate a conversation about CT assessment, by presenting a suite of resources created to help teachers understand a) what computational thinking practices look like when students use a particular programming language (Scratch), b) how to collect examples of students engaging in computational thinking practices, and c) how to assess those examples of student work.

Potential Significance of the Work

Understanding how CT and, more broadly, design thinking align with teachers' and administrators' larger educational goals, and helping them recognize the value of this kind of thinking is critical for innovative, informal computer programming and digital design programs to take hold in formal educational settings.

The Theoretical and Methodological Approaches Pursued

Policy makers and education leaders have recognized the need for young people to have a greater understanding of how computers and other technologies work, how they are designed, and the logical processes upon which computing is based (Guzdial, 2008; Wing, 2006). The CT education research community faces challenges as it advocates for integrating CT into the K–12 curriculum. First, defining a framework for CT remains an ongoing topic of debate (NRC, 2011, 2012). Second, clarifying for the larger education community how knowledge gained from programming and design experiences can transfer to other domains remains a challenge (Han Koh, Basawapatna, Bennett, & Repenning, 2010). Third, developing strategies for assessing CT is critical for adoption in formal school settings, but currently lacks coherence (Brennan, & Resnick, 2012).

The authors were involved in a program to help teachers integrate Scratch into formal instruction. The program developers created professional development opportunities, curricula, and resources for educators. The program researchers were tasked with developing a method for assessing students' development of computational thinking as they used Scratch. This first required a definition of computational thinking by the program developers. Initially, their definition was based on both definitions being formulated in the CT education community and their own experiences seeing young people engage with Scratch in informal settings. The initial CT framework included what they called "computational concepts, practices, and perspectives." Concepts included items such as loops, conditionals, parallelism, etc.; practices included iterating, debugging, abstracting, and reusing; and perspectives included expressing, connecting and questioning.

Using this framework the researchers created protocols for conducting interviews of students talking about their projects and debugging problems in the code. We recorded these using screen capture technology,

which records audio and actions on the computer screen. We interviewed 36 students at the beginning, middle, and end of the units in which they learned Scratch (a total of 91 interviews). We also conducted interviews with twelve teachers at the beginning, middle, and end of their units about their experience integrating Scratch into their instruction, how they were assessing students' Scratch work, and what kinds of assessments and resources they needed to help themselves and their colleagues understand what students were learning with Scratch.

Preliminary Findings

Our analyses of student screen captures and teacher interviews indicated that the most important area to focus on was the CT practices, rather than the concepts or perspectives. Teachers either already were capable of assessing the concepts (if they were computer science teachers) or the concepts were not the main goals of instruction (if they were subject area teachers). Computational perspectives were too varied to operationalize and capture reliably. The CT practices, however, could be operationalized, were relevant for a wide range of teachers, and yet were not generally being assessed because they were about process rather than product. The researchers and developers revised the CT practices framework and developed a range of resources for teachers to use in their instruction, which we piloted with two elementary school teachers, two middle school teachers and one high school teacher. We found through the pilot studies that teachers were able to adapt the assessment materials to a range of teaching situations (high school computer science, middle school technology class, elementary school science), and that students at different age levels were able to complete the assessment materials. The pilot feedback led us to finalize the CT practice framework:

- *Experimenting and Iterating:* Developing a little bit, then trying it out, then developing some more
  This is characterized by the following practices: a) Build a project step by step, b) Try things out as you go, c) Make revisions based on what happens, d) Try different ways to do things, or try new things.
- *Testing and Debugging*: Making sure things work –finding and solving problems when they arise
  This is characterized by the following practices: a) Observe what happens when you run your project, b) Describe what is different from what you want, c) Read through the scripts to investigate the cause of the problem, d) Make changes and test to see what happens, e) Consider other ways to solve the problem.
- *Reusing and Remixing*: Making something by building on existing projects or ideas
  This is characterized by the following practices: a) Find ideas and inspiration by trying other projects and reading the scripts, b) Select an image, sound or script and adapt it for your project, c) Modify an existing project to improve or enhance it, d) Give credit to people whose work you build on or are inspired by.
- *Abstracting and Modularizing*: Exploring connections between the whole and the parts
  This is characterized by the following practices: a) Decide what sprites are needed for your project, and where they should go, b) Decide what scripts are needed for your project, and what they should do, c) Organize the scripts in ways that make sense to you and others.

The teacher resources we developed included: a) Video exemplars of elementary, middle and high school students engaged in each of the practices, b) A teacher self-reflection tool for assessing whether lessons provide opportunities to engage in the practices, c) A set of student reflection questions asking students to show how they engaged in the practices, d) Examples of methods for collecting students' reflections on their practices, e) Examples of students' responses to reflections and sample assessments of the quality of their work.

References

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher 42*(38), 38–43.

Guzdial, M. (2008). Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25–27.

Han Koh, K., Basawapatna, A., Bennett V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. *Proceedings of the 2010 Conference on Visual Languages and Human Centric Computing (VL/HCC 2010)*, 59–66. Spain: IEEE Computer.

National Research Council (2011). *Committee for the Workshops on Computational Thinking: Report of workshop of pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.

National Research Council (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Washington, DC: National Academies Press.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–36.