"Systems of Assessments" for Deeper Learning of Computational Thinking in K-12

Shuchi Grover
Stanford University/SRI International

Please address all correspondence to:
Shuchi Grover
Stanford Graduate School of Education
485 Lasuen Mall, Stanford, CA 94305‐3096
Tel: (650) 723-2109
shuchig@stanford.edu

Draft paper to be presented at the

Annual Meeting of the American Educational Research Association, Chicago, April 15–20, 2015

"Systems of Assessments" for Deeper Learning of Computational Thinking in K-12

Shuchi Grover
Stanford University

Abstract: As K-12 educators move to introduce computing curricula, the issue of assessing student learning of computational concepts remains largely unresolved. This is central, however, if the goal is to help students develop deeper, transferable computational thinking (CT) skills that prepare them for success in future computing experiences. This paper argues for the need for multiple measures or "systems of assessments" that are complementary, attend to cognitive and non-cognitive aspects of CT, and contribute to a comprehensive picture of student learning. It also describes the multiple forms of assessments used in a middle school computing curriculum, including formative assessments such as multiple-choice quizzes, and directed as well as open-ended programming assignments, and summative assessments to measure growth and transfer of CT.

Objectives

"Deeper learning" (Pellegrino and Hilton, 2012) is increasingly seen as an imperative for helping students develop robust, transferable knowledge and skills for the 21st-century. The phrase acknowledges the cognitive, intrapersonal and interpersonal dimensions of learning, while also underscoring the need for learners to be able to transfer learning to future contexts. Ideas of deeper learning find resonance in *How People Learn* (Bransford, Brown & Cocking, 2000)—the seminal treatise that explicated the need for learning environments to be assessment-centered in addition to learner-, knowledge- and community-centered.

Computational Thinking (CT) is widely recognized today as a necessary skill for today's generation of learners (Wing 2006, 2011), and a consensus has been building around the view that all children must be offered experiences with computer science (CS) in their K-12 years (Grover & Pea, 2013). Without attention to assessment, computing can have little hope of making its way successfully into K–12 school education settings at scale.

Using the deeper learning lens for assessing computational learning, this paper reviews the shortcomings in commonly used assessments of CT, and argues for employing "systems of assessments" (Conley & Darling-Hammond, 2013) to assess the development of CT's cognitive and non-cognitive aspects. As a case in point, it describes the systems of assessments employed in an introductory computer science course for middle school students, and the results of empirical inquiries of its use.

Framing the Research

Despite the many efforts aimed specifically at tackling the issue of CT assessment (e.g., Fields, Searle, Kafai & Min, 2012; Ioannidou, Repenning, & Webb, 2010; Meerbaum-Salant et al., 2010; Werner et al., 2012), assessing the learning of computational concepts and constructs in these programming environments remains a challenge. Furthermore, few studies at the K-12 level, if any, have looked at the issue of transfer of CT skills to future learning contexts. New approaches to transfer such as *Preparation for Future Learning* (Schwartz, Bransford & Sears, 2005) have shown promise in the context of science and mathematics learning at the secondary level (Dede, 2009; Schwartz & Martin, 2004). Interventions in CS education could similarly benefit from these emergent ideas from the learning sciences.

In the context of introductory programming, Werner, et al. (2012) conducted a series of investigations involving game programming in Alice with middle school students. Their "Fairy Assessment" requires students to code parts of a pre-designed program to accomplish specific tasks. This assessment is Alice-based, and grading is subjective and time-consuming—a perennial challenge for assessing student code.

Looking at student-created programs alone could also provide an inaccurate sense of students' computational competencies (Brennan & Resnick, 2012). Although time-consuming, "artifact-based interviews" can help provide a more accurate picture of student understanding of their programming projects (Barron et al., 2002). There is a thus a need for more objective assessment instruments as well that can illuminate student understanding of specific computing concepts. Cooper created a multiple-choice instrument for measuring learning of Alice programming concepts (Moskal, Lurie & Cooper, 2004), but it has not been used to measure student learning in K-12 education.

SRI International (2013)'s effort to create systematic frameworks for assessing CT—Principled Assessment of Computational Thinking (PACT)—focuses on assessing CS Concepts, Inquiry Skills, and Communication & Collaboration Skills as key elements of CT practices in the context of the high school ECS curriculum.

Outside the US, introductory computing curricula at the elementary and middle school levels in countries such as the UK (Scott, 2013) and Israel ((Zur Bargury, 2012) also provide useful ideas for assessment. The Israeli effort uses multiple-choice assessments and attendant rubrics (Zur Bargury, Pârv & Lanzberg, 2013) that make it easier to measure learning in a large-scale setting than open-ended student projects.

Barron and Darling-Hammond (2008) contend that robust assessments for meaningful learning must include: (1) intellectually ambitious performance assessments that require application of desired concepts and skills in disciplined ways; (2) rubrics that define what constitutes good work; and (3) frequent formative assessments to guide feedback to students and teachers' instructional decisions. Conley and Darling-Hammond (2013) assert that assessments for deeper learning must measure:

1. Higher-order cognitive skills, and more importantly, skills that **support transferable learning,** and
2. Abilities such as collaboration, complex problem solving, planning, reflection, and communication of these ideas through use of appropriate vocabulary of the domain in addition to presentation of artifacts to a broader audience. These assessments are in addition to those that measure key subject matter concepts.

This assertion implies the need for multiple measures or "systems of assessments" that are complementary, encourage and reflect deeper learning, and contribute to a comprehensive picture of student learning. Few to none of the prior efforts described above nor current computing curricula (such as those promoted by Code.org or Khan Academy) include such comprehensive assessments, if they assess at all.

Methods

The remainder of the paper describes the second iteration of a design-based research effort concerning the use of 'Foundations for Advancing Computational Thinking' (FACT), a six-week introductory CS course designed and created on the OpenEdX online platform for blended in-class learning in middle school. FACT included elements designed to build awareness of computing as a discipline while promoting engagement with foundational computational concepts such as algorithmic flow of control: sequence, looping constructs, and conditional logic. The course comprised short Khan Academy-style videos ranging between 1-6 minutes in length that led learners through the thinking and construction of computational solutions using the Scratch programming environment. The videos were interspersed with programming activities to be completed individually or in pairs, and quizzes that used automated grading and provided feedback.

The goal of the research was to study multiple and novel "systems of assessments" to measure growth of algorithmic thinking skills and transfer of these skills, as well as non-cognitive aspects such as perceptions of computing, and communication of CT ideas. In light of the recommendations above, multiple and innovative measures of assessment were used in FACT to help create a multi-faceted picture of student learning as described in the sections below.

FACT's Systems of Assessments

Well-designed multiple-choice assessments can be used to further learners' understanding (Glass & Sinha, 2013), and provide learners with feedback and explanations rather than simply testing (Black & William, 1998). Low-stakes, high-frequency quizzes throughout FACT tested students' understanding of specific CS concepts and constructs, and gave learners immediate feedback on their understanding. This feedback could prompt learners to re-watch the video lecture, thus allowing them to take more control of their learning. Many quizzes involved small snippets of Scratch or pseudo-code on which questions were based (Figure 1). These assessments were designed to help learners develop familiarity with code tracing–the ability to read/understand code (Bornat, 1987; Lopez, Whalley, Robbins, & Lister, 2008). Some quiz questions also involved presenting jumbled blocks in Scratch required for a program and having students snap them in correct order. This type of question was inspired by Parson's puzzles (Denny, Luxton-Reilly, & Simon, 2008; Parsons & Haden, 2006).

FACT placed a heavy emphasis on 'learning by doing' in the Scratch programming environment. In addition to open-ended time to dabble with programming following videos, there were specific assignments with attendant rubrics that built on the concepts taught in the videos (Table 1). Rubrics included items for creativity that encouraged students to add their own distinctive elements.

Summative assessments involved a final exam online that included multiple-choice and open-ended questions that assessed learners' CT ability through questions that required code-tracing and/or debugging (Figure 2). They also included questions from the 2012 Israel National Exam in order to conduct a comparative analysis with the results

reported in Israel (Zur-Bargury, Parv & Lanzberg, 2013). A final project of learner's choosing to be done with a partner was also included as part of summative assessments. In keeping with desired social and participatory aspects of learning environments, the final projects were presented during a whole-class 'Expo', and were showcased in an online studio of games on the Scratch site so classmates could play with, and provide feedback on, their peers' games. The different tasks involved creating the project, testing it, demonstrating it to the entire class, documenting and writing reflections on it, and finally, playing with each others' games in the online studio. These afforded learners the opportunity to problem solve, collaborate, plan, communicate, present, and reflect on their work in a document adapted from the *Starting from Scratch* curriculum (Scott, 2013). Inspired by past research (Barron et al., 2002), "artifact-based interviews" explicating their Scratch projects were conducted.

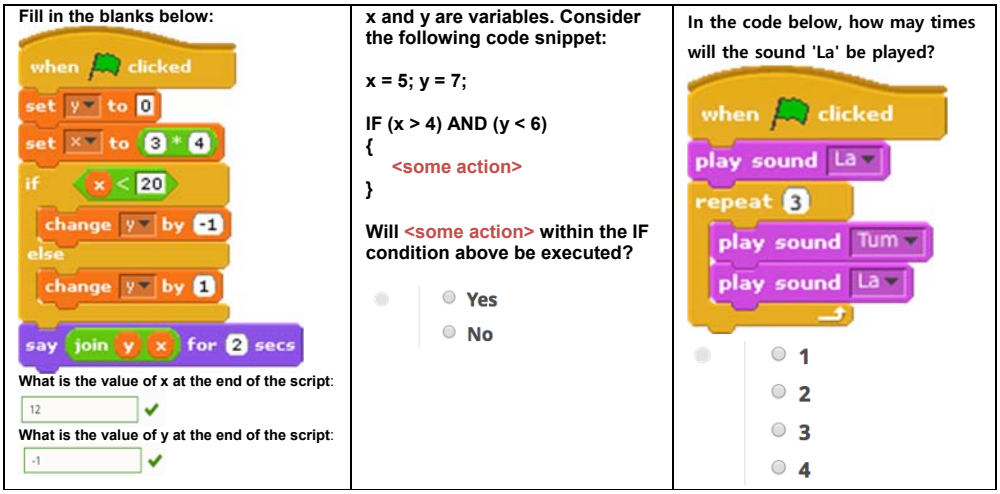

Figure 1: Sample quiz questions used in formative assessments

Table 1. Sample structured and open-ended Scratch programming assignments in FACT

Programming Assignments (Scratch/Pseudocode)	**Algorithmic/CT concepts/constructs**
Share a recipe	Sequence of instructions– serial execution; repetition; selection
(Scratch) Make a life cycle of choice to use in a 6th grade science class	Serial execution
(Scratch) Draw a Spirograph from a polygon of choice	Simple nested loop + creative computing
(Scratch) Create a simple animation of student's choice	Forever loop
(Scratch) Generic polygon maker	Variables; user input
Look inside Scratch code and explain the text version of code	Algorithms in different forms (analogous representations for deeper learning)
(Scratch) Draw a 'Squiral'	Loops, variables, creative computing
Open-ended project (in pairs): Create a game using "Repeat Until"	*Loops ending with Boolean condition*
(Scratch) Maze game	Conditionals; event handlers
(Scratch) Guess my number game	Loops, variables, conditionals, Boolean logic
(Scratch) Final, open-ended project of choice	*All CT topics taught in FACT*

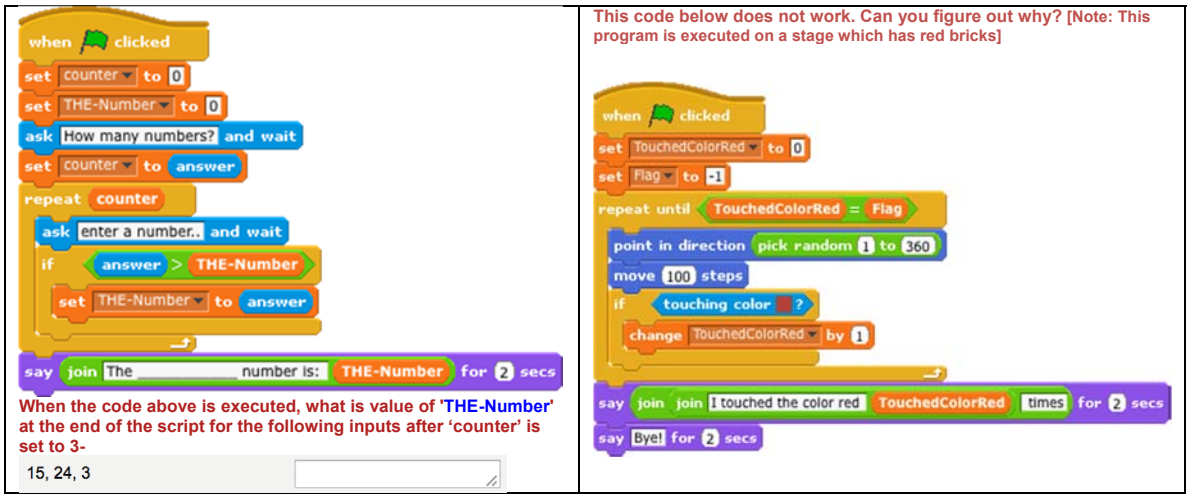

Figure 2. Samples questions in the CT summative test (in addition to questions from Israel Nationwide Exam)

Another unique aspect of FACT was the 'Preparation of Future Learning (PFL) Test' that was specially designed and administered after the end of the course to assess how well students were able to transfer their computational understanding built in the block-based Scratch programming environment to the context of a text-based (Java-like) programming language.

Lastly, since perspectives and practices of the discipline are key ingredients of good STEM curricula, affective aspects such as students' growth in their understanding of computing as a discipline and changes in their attendant attitudes towards CS were also assessed through pre-post responses to the free-response question– "What do computer scientists do?"

Participants & Procedures

FACT was taught in a public middle school classroom in Northern California. The student sample comprised 28 children from 7th and 8th grade (20 boys and 8 girls, mean age: ~12.3 years) enrolled in a semester-long "Computers" elective class. The class met for 55 minutes four times per week. The classroom teacher and researcher were present in the classroom at all times. IRB permission was sought from parents and students prior to the start of the study. An independent researcher assisted with grading.

Data Measures

- *Prior Experience Survey*: These gathered information about students' prior experiences in computational activities.
- *Pre-Post CS Perceptions survey*: Including free-response question: "*What do computer scientists do?*".
- *Pre-Post CT Tests***:** This measured pre- and post-FACT CT skills.
- *Quizzes:* These monitored student progress and captured conceptual targets of difficulty.
- *Scratch Assignments:* 10 assignments throughout the course (graded per rubrics).

- *Final Scratch Projects, Presentations & Student Interviews on the final project.*
- "*Preparation of Future Learning*" (PFL) test designed to assess transfer of learning to a text-based programming language.

Results

The pretest-to-posttest effect size (Cohen's *d*) was ~2.4, and all learners showed statistically significant learning gains on the CT Test (Tables 2 & 3). Pretest scores were strong predictors of performance on the posttest. The comparative analysis with the results from the Israeli nationwide exam revealed comparable or slightly better performances by our students (Figure 3). The average score for the 10 Quizzes was 73.14 (S.D. 12.68).

Table 2. Within-Student Comparison of Pretest and Posttest Scores (Cohen's *d = 2.4)*

	Pretest Score		Posttest Score			
N	Mean	SD	Mean	SD	t-stat	p-value
28	28.1	21.2	81.6	21.0	-15.5	<0.001

The t-stat and its following p-value come from a t-test to test whether the pre and post means are the same.

Table 3. Posttest Scores Breakdown by CS Topics

	Mean	Std. Dev.
Overall Score	81.6	21.2
By CS Topic		
Serial Execution	91.1	20.7
Conditionals	84.9	20.5
Loops	77.2	26.3
Vocabulary	77.4	22.2

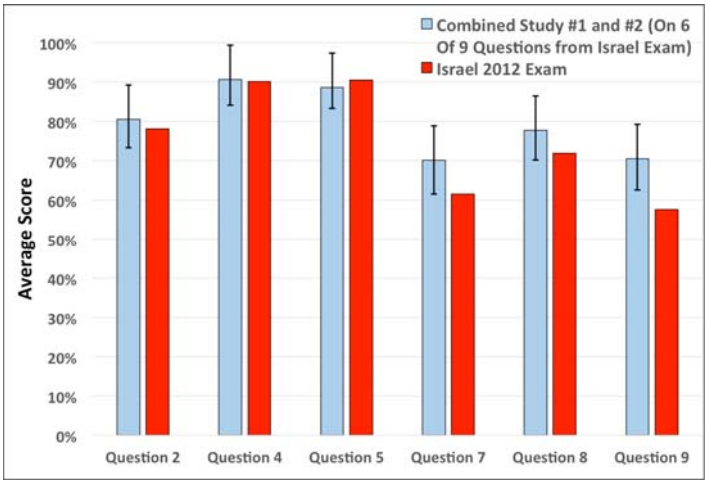

Figure 3: Student performance post-FACT vs. 2012 Israel National Exam results (N=4082)

On the PFL test, although students scored an average of 65%, there was evidence of understanding of algorithmic flow of control in code written in a text-based programming language. Regression analyses revealed that ELL students had trouble with the text-heavy PFL test. Qualitative analyses of the final projects (Figure 4) and student interviews revealed that even the students who had performed poorly in the posttest and

PFL test showed high levels of engagement in the final project as well as a reasonable understanding of algorithmic constructs (Table 4).

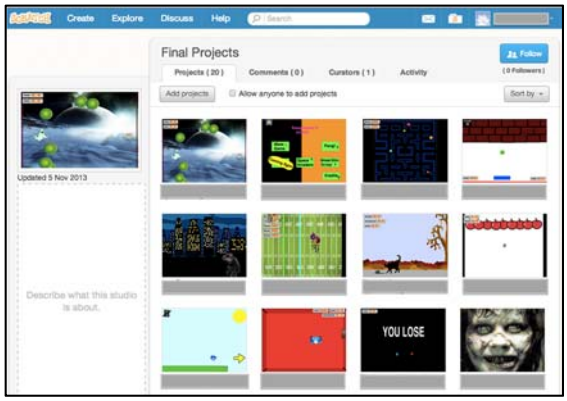

Figure 4. Studio of Students' Final Projects on Scratch website

Students' pre-post responses to the question "What do computer scientists do?" revealed a significant shift from naïve "computer-centric" notions of computer scientists as people who engage mostly in fixing, making, studying or "experimenting" with computers. Their perceptions of computing matured to embrace a view of CS as an engaging problem-solving discipline with diverse real-world applicability (Figure 5).

Table 4. Highlights of "Artifact-Based" Interview with Kevin (pseudonym), among the lower performers on the CT posttest and PFL test

Observation/Interpretation	***Student Quote(s) or Exchanges with Interviewer***
Connected "scary" final project idea with event in personal life	*"we watched **Insidious** the day before… and then we got this project assigned and we just decided to do something scary.. coz we were traumatized coz of the movie*
Decision to do Maze game, (but with conscious effort to make enhancements)	*"We decided to do a maze because, uhm, well, we had an assignment where we were supposed to do a maze.. before… We also added like uhm, like levels and stuff like that."*
Able to describe what the code did, referred to specific code elements while talking.	*e.g. "we have in forever our up and arrow keys..."* *"And each level has a different color red; so when you touch it it'll take you to the next level"*
Was aware of a bug in the code and showed it.	*Here you see when you touch the black, it'll start you down here, instead of up there.. and w-we don't know why. We tried to fix it but.. we couldn't fix it, I don't know what was going on..*
Talked through problem behind bug, got to a solution when guided. He figured an IF-ELSE would be needed to know which level they were on and reset position accordingly.)	

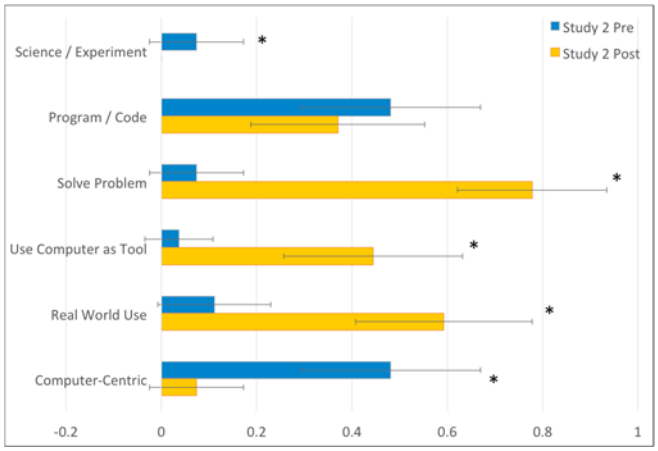

Figure 5. "What do computer scientists do?" Pre-Post FACT intervention responses

Conclusions & Implications

No single one of the FACT assessments alone would have assessed the cognitive, affective and transfer aspects of deeper learning. Together, however, these assessments provide a more comprehensive view of student learning than commonly used assessments of CT such as a student-created program. This research thus argues for "systems of assessments" to measure deeper learning and demonstrates these for a structured introductory computing course. It addresses a crucial lacuna in what has traditionally served as assessment in prior CS education research in K-12. This is especially relevant today as computing inexorably makes way into K-12 classrooms. Future work includes validating these assessments across learners and contexts.

We gratefully acknowledge NSF for their funding (Grant # _____).

References

Barron B., & Daring-Hammond, L. (2008). How can we teach for meaningful learning.? In Daring-Hammond, L., Barron, B., Pearson, P. D., Schoenfeld, A. H., Stage, E. K., Zimmerman, T. D., Cervetti, G. N., & Tilson, J. L. *Powerful learning: What we know about teaching for understanding*. San Francisco: Jossey-Bass.

Barron, B., Martin, C., Roberts, E., Osipovich, A., & Ross, M. (2002, January). Assisting and assessing the development of technological fluencies: Insights from a project-based approach to teaching computer science. In *Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community* (pp. 668-669). International Society of the Learning Sciences.

Bransford, J. D., Brown, A., & Cocking, R. (2000). How people learn: Mind, brain, experience and school, expanded edition. *DC: National Academy Press, Washington*.

Brennan, K., & Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking.

Bornat, R. (1987). *Programming from First Principles*. Prentice Hall International.

Black, P., & Wiliam, D. (1998). *Inside the black box: Raising standards through classroom assessment*. Granada Learning.

Conley, D. T., & Darling-Hammond, L. (2013). Creating Systems of Assessment for Deeper Learning.

Dede, C. (2009). Immersive interfaces for engagement and learning. *Science, 323*(5910), 66-69.

Denny, P., Luxton-Reilly, A., & Simon, B. (2008, September). Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on Computing education research* (pp. 113-124). ACM.

Fields, D. A., Searle, K. A., Kafai, Y. B., & Min, H. S. (2012). Debuggems to assess student learning in e-textiles. In *Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education*. New York, NY: ACM Press.

Glass, A. L., & Sinha, N. (2013). Providing the answers does not improve performance on a college final exam. *Educational Psychology*, *33*(1), 87-118.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43.

Ioannidou, A., Repenning, A., & Webb, D. C. 2009. AgentCubes: Incremental 3D end-user development. *Journal of Visual Languages & Computing*, *20*(4), 236-251.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008, September). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth international Workshop on Computing Education Research* (pp. 101-112). ACM.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M., (2010). Learning computer science concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, 69-76.

Moskal, B., Lurie, D., & Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. *ACM SIGCSE Bulletin, 36*(1), 75-79.

Parsons, D., & Haden, P. (2006). Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 157-163). Australian Computer Society, Inc.

Pellegrino, J. W., & Hilton, M. L. (Eds.). (2013). *Education for life and work: Developing transferable knowledge and skills in the 21st century*. National Academies Press.

Schwartz, D. L., Bransford, J. D., & Sears, D. (2005). Efficiency and innovation in transfer. In J. Mestre. (Ed.), *Transfer of learning from a modern multidisciplinary perspective* (pp. 1-51). Greenwich, CT, Information Age Publishing.

Schwartz, D. L. & Martin, T. (2004). Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. *Cognition and Instruction*, *22(2)*, 129-184.

Scott, J. (2013). The royal society of Edinburgh/British computer society computer science exemplification project. *Proceedings of ITiCSE'13*, 313-315.

SRI International (2013). *Exploring CS Curricular Mapping*. Retrieved from http://pact.sri.com/?page_id=1380

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The Fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*, 215-220. New York, NY: ACM.

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

Wing, J. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*, Spring. Carnegie Mellon University, Pittsburgh. Retrieved from http://link.cs.cmu.edu/article.php?a=600

Zur Bargury, I. (2012). A new curriculum for junior-high in computer science. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* 204-208. ACM.

Zur Bargury, I., Pârv, B. & Lanzberg, D. (2013). A Nationwide Exam as a Tool for Improving a New Curriculum. Proceedings of ITiCSE'13, 267-272. Canterbury, England, UK.