Designing An Online Computer Science Course for Deeper Computational Learning in Middle School

SRI Education

Shuchi Grover, SRI Education; Roy Pea, Stanford University

Stanford University

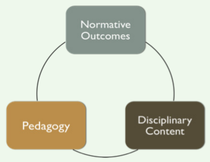
- Middle school introductory CS course on Stanford OpenEdx
  - Focus on deeper learning
    - Core concepts & transferable skills (Cognitive)
    - Interest & Awareness of CS (Affective)
  - 'Balanced' pedagogy for deeper learning
    - Preparation for Future Learning (PFL)
    - Remedying Misperceptions of Computing
  - "System of Assessments"
  - MOOC with a difference created using Design-Based Research (DBR)

Motivation

The middle school years are formative and key for cognitive and social development in the K-12 schooling journey especially with regard to future engagement with STEM+C fields (Tai et al., 2006). Middle schools across the US face an acute shortage of teachers to teach introductory computing. Making available a robust, structured curriculum as an online course would accelerate scaling to wider audiences of students and teachers. The recent explosion of MOOC platforms and courses could serve this need in K-12.

Research Framework

Programming and computational problem solving is difficult for novice learners (du Boulay, 1989; Pea & Kurland, 1984; Robins, Rountree, & Rountree, 2003). "Deeper learning" (Pellegrino and Hilton, 2012) helps students develop robust, transferable knowledge and skills for the 21st-century. It acknowledges the cognitive, intrapersonal and interpersonal dimensions of learning, while also underscoring the need for learners to be able to transfer learning to future learning contexts. The pedagogical framework draws on research in how people learn, and how to help novice programmers learn the desired disciplinary content this curriculum seeks to teach.



Methods

This preliminary DBR (Barab & Squire, 2004) effort presents results of two iterations of a seven-week course titled "Foundations for Advancing Computational Thinking" (FACT) used in a blended classroom setting in a public middle school. The second iteration involved investigations on an online version of FACT designed and created on the OpenEdX MOOC platform. The empirical inquiry aimed to answer the following research questions:

1. What is the variation across learners in learning of algorithmic flow of control (serial execution, looping constructs, and conditional logic) through the FACT curriculum?
2. Does the curriculum promote an understanding of algorithmic concepts that goes deeper than tool-related syntax details as measured by Preparation for Future Learning (PFL) assessments?
3. What is the change in the perception of the discipline of CS as a result of FACT?

Foundations for Advancing Computational Thinking

Pedagogy For Deeper Learning

- "Expansive framing" – deeper structure applicable to all languages
- Worked examples – instruction in a problem-solving context
- Use of pseudo-code – planning + analogous representation
- Modeling reading/code-tracing (computing 'practices')
- Academic language to explain CS concepts (computing 'practices')
- Frequent multiple-choice "quizzes", including Parson's puzzles
  - Immediate feedback; push understanding; expose gaps
  - Low stakes – high frequency
- Several hands-on programming activities and assignments ('creativity' item in rubric)

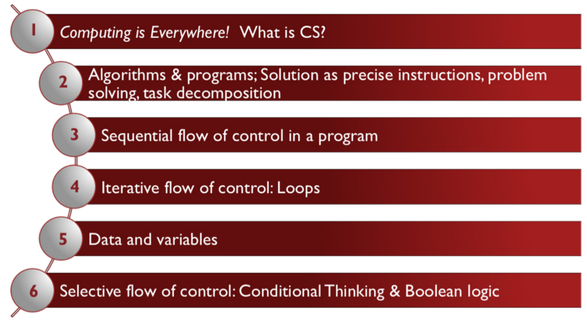
Pedagogy For Active Learning

- Short videos
- Start with a problem (= set up context ➜ priming for learning)
- End a video with a question
- Discussion / Scratch window / Problem Set below video
- Relevant contexts (art, games, animations)
- Open-ended questions / reflections
- Activities involving elbow partner / Pair projects / Open programming day / Students helping students catch up
- Final Project – Whole-class demo / Online showcase of games

Curriculum Design

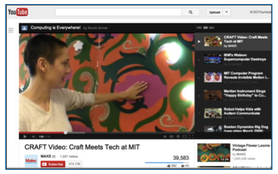
7-week "ECS-lite" curriculum focusing on--

- The role of computing in our lives and applications of computation.
- Algorithmic problem solving.
  - Building blocks of all algorithmic solutions, namely, sequence of instructions, serial execution, repetition, and conditional logic (including Boolean operators).
- Gaining a familiarity with the vocabulary of computer science
- Learning core practices of programming, including:
  - Task decomposition and composition of a programmatic solution.
  - Using abstraction including algorithmic solutions as pseudo-code.
  - Iterative development of programs.
  - Debugging and testing of programs.
- Interpersonal and intrapersonal aspects of computational learning:
  - Working in collaboration with others.
  - Learning from and supporting one another's learning.
  - Showcasing and sharing one's work products in the classroom community.
  - Reflection and revision.



Perceptions of Computing

Curation of publicly available videos showcasing computing in action in exciting ways (http://bit.ly/CS-rocks) and creation of a corpus of video vignettes of diverse people describing how they use computing for their professions and passions (http://bit.ly/CS-video-vignettes).



Sample and Data Measures

		Count by Gender		Count by Grade		Count in Sp. Programs	
Study	Mean Age	Male	Female	Grade 7	Grade 8	ELL	Special Ed.
1	12.9	21	5	15	11	4	2
2	12.3	20	8	16	12	3	1

Instrument	Pre-Intervention	Post-Intervention	Source(s)
Computational Knowledge Test	✓	✓	Inspired by Ericson and McKlin (2012); Meerbaum-Salant, et al. (2010); Zur Bargury, Pârv and Lanzberg (2013)
Preparation for Future Learning (PFL) Test		✓	Designed (Inspired by Schwartz and Martin (2004))
Prior Experience Survey (programming experience and technology fluency)	✓		Adapted from Barron (2004)
CS Perceptions Survey	✓	✓	Ericson and McKlin (2012) and open-ended questions like "In your view, what do computer scientists do"
Online Learning Experience Survey (Only Study 2)	✓	✓	Designed (Inspired by depth and breadth of experience items as in Barron (2004))
FACT Experience survey		✓	Designed (for getting student feedback for future improvements)
"Artifact-based interviews" on final project		✓	

Results

Gender differences were observed with girls performing better than boys, however the small number of females in the sample precluded drawing deeper conclusions. **Regression analyses suggested that the curriculum helped all students regardless of prior experience as measured by the self-report survey, however the pretest score and prior math ability (as measured by CA state tests) was found to be significant predictors for both the posttest and the PFL test score.**

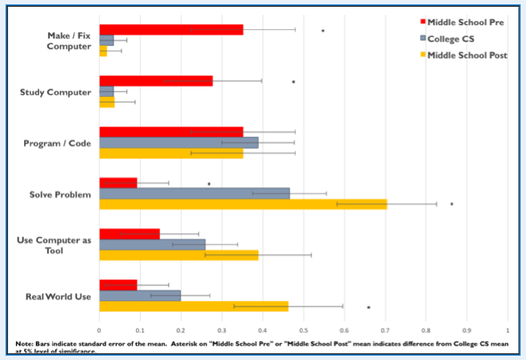
Comparison of Scores (out of 100) of Computational Knowledge and PFL Tests, Study 1 & Study 2

	Study 1		Study 2					
	N	Mean (SD)	N	Mean (SD)	t	p < |t|	z	p < |z|
Pretest	24	36.33 (18.19)	28	28.06 (21.18)	1.5	0.14	1.76	0.08
Posttest	26	78.58 (17.08)	28	81.60 (21.24)	-0.58	0.56	-1.26	0.21
Learning Gain	24	43.08 (12.17)	28	53.07 (18.34)	-2.34	0.02*	-2.46	0.01*
PFL Test	25	63.37 (28.86)	27	65.07 (26.47)	-0.22	0.82	-0.08	0.93

Posttest Scores by CT Topics, Study 1 & Study 2

	Study 1	Study 2				
Variable	Mean (SD)	Mean (SD)	t-Stat	p	Z-Score	p
Overall	78.6 (17.1)	81.6 (21.2)	-0.6	0.56	-1.3	0.21
By CS Topic						
Serial Execution	97.4 (13.1)	91.1 (20.7)	1.4	0.18	1.6	0.12
Conditionals	84.5 (19.0)	84.9 (20.5)	-0.1	0.94	-0.4	0.72
Loops	74.1 (21.9)	77.2 (26.3)	-0.5	0.64	-1.1	0.29
Vocabulary	68.2 (17.9)	77.4 (22.2)	-1.7	0.10	-2.0	0.05*

"What do computer scientists do?"- College students' responses (N=168) vs. pre-post responses from Study 1 & Study 2 (combined)



Highlights of "Artifact-Based" Interviews

Artifact-based interviews with students who were among the lower performers on the CT posttest and PFL test revealed that students' final projects evinced high levels of engagement and showed evidence of understanding of program construction and algorithmic constructs that the posttest was unable to capture.

This material is based upon work supported by the National Science Foundation under NSF No. 1343227. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.