Emerging Design Principles for Curriculum to Integrate Computer Coding into Middle School Mathematics

Jessica L. Smith, Ellen Granger, Sherry A. Southerland, Christy Andrews-Larson, Xin Juan, David Whalley, Matthew Mauntel, Shafayat Rahman, & Yasser Atiya

School of Teacher Education, FSU-Teach, Office of Science Teaching Activities and Outreach, & Computer Science

This is part of a larger effort bringing together a diverse design team, to create curriculum integrating computer coding and middle-school general-mathematics. The goal was to enhance the instruction of students that have been traditionally underserved in mathematics by using computer science ideas found in coding to complement, reinforce, and build on mathematics ideas in a meaningful way. The development of modules was guided by the principles of Design-Based Research and Realistic Mathematics Education instructional design heuristics, in particular by drawing on the notion of *guided reinvention* through *emergent models*. Here we present the design principles that emerged from the first half of the effort with the hopes of informing other projects that integrate coding and mathematics learning.

Paper presented the annual meeting of the American Educational Research Association, Toronto, Canada, April, 2019.

Emerging Design Principles for Curriculum to Integrate Computer Programming into Middle School Mathematics

Though an overwhelming majority of parents surveyed express a desire for computer science (CS) offerings in their child's school, less than 75% of U.S. K-12 schools offer high quality programming courses (The White House, 2016). Further, many states do not currently require computer science courses to count toward high school graduation—though this does appear to be rapidly changing. As an additional challenge, there is a lack of diversity in students currently engaged in CS. For example, only 18% of CS undergraduates were women in 2012, the most recent year for which NSF data are available (Ashcraft et al, 2012). For minorities the picture is even bleaker—for example, less than 4% of CS undergraduate degrees were awarded to African Americans in 2010-11 (Zweban, 2012). This is mirrored in U.S. technology firms in which women compose less than one-third of the technical workforce and African Americans less than 3% (The White House, 2016). To begin to reverse these trends, students must be engaged at an early point in their academic paths, at a time when they are developing their conception of themselves as STEM "able" or not. Middle school has been shown to be a critical time for this development for science (e.g., Osborne et al., 2003) and mathematics (e.g., Finger & Silverman, 1966; Lipsitz, 1980; Oppong, 2002); thus, it is reasonable to pose middle school as also important for CS (Knezek, Christensend, Tyler-Wood, & Periathiruvadi, 2013). Given recent accountability pressures faced by schools, we argue that it is strategic to use CS to develop students' understanding of mathematics topics. Finally, as there are growing national, state, and district interests in CS integration with mathematics, the determination of school and teacher capacity for CS instruction is critical for the success of such efforts. The research presented here is part of a larger effort that brought together a design team, composed of middle-school mathematics teachers and university CS and STEM education faculty, to design, develop, and test modules in which CS is integrated into the instruction of middle-school general-mathematics courses.

Objective

The objective of this part of the larger research effort is to identify a set of design principles that emerged from curriculum development team as they constructed modules in which computer programming is integrated into mathematics instruction. The overall goal for the modules was to enhance the instruction of students that have been traditionally underserved in mathematics instruction and to introduce them to CS programming. To achieve this goal we use CS ideas found in programming in a meaningful way to complement, reinforce and build on mathematics ideas.

Theoretical Framework

The development of modules in this project was guided by the principles of Design-Based Research (Brown & Campione, 1996; Design-Based Research Collective, 2003). The overall goal of Design-Based Research is to create and extend knowledge about the development, enactment, or methods used to sustain an innovative learning environment, curriculum, or instructional model. Because it is iterative in nature, Design-based research also enables researchers to progressively hone an innovation while building a theory about how it works. According to the Design-Based Research Collective (2003), good design-based research includes the following characteristics:

- Takes place through one or more iterative cycles of design, enactment, analysis, and redesign;
- Incorporates data gathering techniques that can document and connect processes of enactment to outcomes of interest;
- Documents success or failure in authentic settings and focuses on the interactions that refine our understanding of the learning issues involved; and,
- Leads to sharable design principles or strategies that help communicate relevant implications to practitioners and other educational designers (p. 5).

Realistic Mathematics Education (RME) instructional design heuristics have informed the development of modules in this project, in particular by drawing on the notion of *guided reinvention* through *emergent models* (Gravemeijer & Doorman, 1999). This approach begins by positioning students to develop their own informal strategies to solve problems set in an initial context that is meaningful to students; this context might be a real-world setting or a mathematical setting that is familiar to students. The strategies students develop in this context become the basis for subsequent mathematical activity, as students move to generalize their strategies by applying them to similar contexts and eventually formulate descriptions of how their strategies might be used more generically across a set of examples or contexts. Mathematical generalizing requires one to develop symbolism that captures the ideas being examined, symbols that connect to the language and notation of the broader mathematical community; for this, the teacher plays an important role in brokering this relationship (Rasmussen Zandieh, & Wawro, 2009). This approach to instructional design aligns well with our efforts to teach mathematical topics through computer programming, as students' work writing programs creates both a need for and a tool to support generalization. This work will create opportunities to examine the plausibility of this alignment as well as ways in which mathematical symbolization might be both symbiotic and at times in tension with symbolism used in programming.

We draw on a communities of practice lens for conceptualizing the research entailed in this project (Wenger, 1998). In particular, we think of this project as bringing together members of several communities: middle school mathematics teachers, computer scientists, and STEM education researchers. This lens is fruitful for conceptualizing the joint work of these groups in designing and refining instructional modules, as well as conceptualizing the work teachers do implementing them with students in classrooms. In the context of mathematics classrooms informed by RME instructional design, teachers play an important role in brokering between students' mathematical activity and the language, notation, and ways of reasoning commonly held by the broader mathematics community (e.g. Rasmussen, Zandieh, & Wawro, 2009).

The work of developing the capacity to teach mathematics through computer science necessitates what Akkerman and Bakker (2011) refer to as "boundary crossing" for members of different communities involved in the curriculum design work. For instance, the computer scientists are pressed to consider how children learn and how teachers engage in their professional work.

Mathematics educators are pressed to consider how children's mathematical learning will be reshaped through the integration of programming and the use of a programming platform. Mathematics teachers are pressed to coordinate the new tool of a programming platform with their experiential and professional knowledge of teaching middle grades mathematics. This boundary crossing is a goal-driven one and has resulted in the development of instructional modules that function as boundary objects which "fulfill a specific function in bridging intersecting practices" (Akkerman & Bakker, 2011) of these communities.

Methods

Research Design Overview. The goal of the larger project is to develop modules to engage students in mathematical experiences through computer programming; thus, mathematics and CS instruction is to be organized around the same math content that is already part of the curriculum. The project plan of the larger effort included three development, enactment, analysis, and redesign cycles. The initial module development occurs in summer institutes followed by in-school cycles of module enactment and analysis, with redesign occurring the following summers. (See Figure 1). (Note: The results of years 1 and part of year 2 are reported here.) The project consists of two iterative cycles of module design, classroom enactment, analysis, and redesign (Design-Based Research Collective, 2003) followed by a comparative pilot test (Year 3).

For each of these cycles, module development data is collected before, during, and after the summer design phase and academic year enactment phase. Likewise, for each of these cycles student data are collected before, during, and after academic-year module enactment. This iterative cycling enabled the team to progressively refine the modules and identify principles for their design. The design principles presented here emerged from the work of the design team to date. The sequence for the design work is described in Table 1 and the modules developed in those years can be found in Table 2.

Participants. The design team consists of 11 middle school mathematics teachers, 2 computer scientists, 2 computer science graduate students, a mathematics educator, 2 mathematics education graduate student, and 2 STEM teacher educators. Each teacher taught 1-2 sections of general mathematics (or more depending on school size and teacher assignment), with 20-25 students per class. Demographics of the teachers' schools are described in Table 3.

Data Sources & Analysis

Data employed in this work include that gathered from the summer institutes as well as data drawn from the enactment of the modules during the academic year. The summer institute data included: (1) video and audio recordings of development sessions, (2) collection of development work products: photos of white board sessions and drafts of modules, and (3) interviews with design team members to determine the principles that drive their work in the development sessions, their ongoing hopes and concerns about the materials being produced, and the factors that most influence their design choices. Data from the classroom enactment of the modules include: (1) video and audio recordings of students engaged in the modules, (2) student work products arising from the modules—including programming in SCRATCH, and (3) interviews with a purposeful sampling of students (traditionally high/medium/low achieving students) from each class. Post mathematics and CS assessment data was also examined for possible contributions to this question.

Findings

The design principles emerging from the first two years of the project can be found in Table 4. These principles are posed as questions to be used by a design team to query the curriculum as it is developed. As shown in this table, many of the design principles focus on the development of a coherent mathematical story line as well as a logical progression of computer programming ideas, as the initial year of field testing indicated that it was common for teachers to lose track of development of one of these as they taught. Other design principles emerged when it became clear that the modules developed, while motivated by teachers' interest in the programming platform (Scratch), quickly grew in length beyond a reasonable timeframe for the classroom. Related to this, the design principles that focus on length and identification of a clear conceptual target emerged from the same pattern. In the full paper, the empirical motivation for each of these design principles will be explored.

Significance

The outcomes of this exploratory research and development have the potential to impact society more broadly by providing a set of design principles that can guide the development of modules in which difficult mathematics concepts are taught through the use of computer programming. Through research on the effectiveness of computer programming in supporting mathematics learning at a critical stage in mathematical development (middle school) these design principles have the potential to contribute to an increase in the number of students interested in and prepared for high-school and post-secondary academic work in computer science and mathematics fields. As importantly, the use of these design principles to create other modules in which computer programming is integrated with mathematics may allow for a wider segment of students to become productively engaged in both mathematics and computer programming, as early introduction to meaningful, problem-driven work in both disciplines will allow for the cultivation of interest and development of proficiency. We hope that such a foundation will be instrumental in broadening the representation of girls and minorities in CS and STEM fields.

References

Akkerman, S. F., & Bakker, A. (2011). Boundary Crossing and Boundary Objects. *Review of Educational Research*, *81*(2), 132–169.

Ashcraft, C., Eger, E., Friend, M. (2012) Girls in IT: The Facts. The National Center for Women and Information Technology. https://www.ncwit.org, Retrieved 2/29/2016

Brown, A. L., & Campione, J. C. (1996). Psychological theory and the design of innovative learning environments: On procedures, principles, and systems. In L. Schauble & R. Glaser (Eds.), *Innovations in learning: New environments for education* (289-325). Mahwah, NJ: Lawrence Erlbaum Associates.

Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, *32*, 5-8.

Finger, J.A., Silverman, M. (1966) Changes in academic performance in the junior high school. *Personnel and guidance Journal*, 45:157-164.

Gravemeijer, K., & Doorman, M. (1999). Context problems in realistic mathematics education: A calculus course as an example. *Educational studies in mathematics*, *39*(1-3), 111-129.

Knezek, G., Christensen, R., Tyler-Wood, T., & Periathiruvadi, S. (2013). Impact of environmental power monitoring activities on middle school students' perceptions of STEM. *Science Education International*, 24(1), 98- 123.

Lipsitz, J.S. (1980) The age group. In M. Johnson (ed.) *Toward adolescence: Seventy-ninth Yearbook of the National Society for the Study of Education.* (pp. 7-31). Chicago: University Press of Chicago.

Oppong, A.K. (2012) Secondary students' attitudes towards mathematics. *Ife Psychologia*, Vol. 20. https://www. Questia.com.

Osborne, J., Simon, S., & Collins, S. (2003). Attitudes towards science: A review of the literature and its implications, International Journal of Science Education, 25(9), 1049-1079,

Rasmussen, C., & Marrongelle, K. (2006). Pedagogical content tools: Integrating student reasoning and mathematics in instruction. *Journal for Research in Mathematics Education*, 388-420.

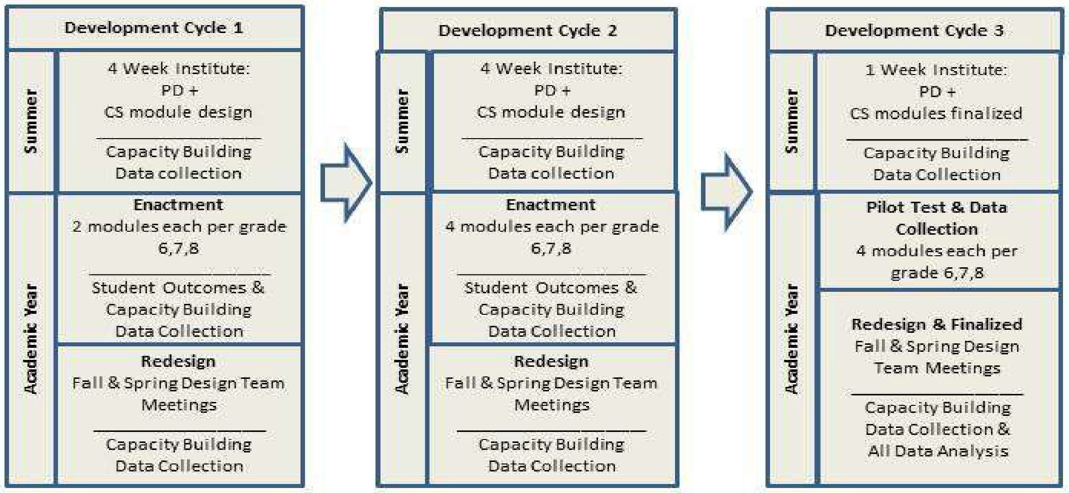
Rasmussen, C., Zandieh, M., & Wawro, M. (2009). How do you know which way the arrows go? The emergence and brokering of a classroom mathematics practice. *Mathematical representations at the interface of the body and culture*, 171-218.

The White House, 2016, https://www.whitehouse.gov, retrieved 3/1/2016.

Wenger, E. (1998). *Communities of Practice*. Cambridge, UK: Cambridge University Press.

Zweban, S. (2012) Computing Degree and Enrollment Trends. The Computing Research Association. *archive2.cra.org, Retrieved 3/1/2016*

Figure 1. Module and Research Development Cycles



Development Cycle 1
Summer
4 Week Institute: PD + CS module design
Capacity Building Data collection
Academic Year
Enactment
2 modules each per grade 6,7,8
Student Outcomes & Capacity Building Data Collection
Redesign
Fall & Spring Design Team Meetings
Capacity Building Data Collection
Development Cycle 2
Summer
4 Week Institute: PD + CS module design
Capacity Building Data collection
Academic Year
Enactment
4 modules each per grade 6,7,8
Student Outcomes & Capacity Building Data Collection
Redesign
Fall & Spring Design Team Meetings
Capacity Building Data Collection
Development Cycle 3
Summer
1 Week Institute: PD + CS modules finalized
Capacity Building Data Collection
Academic Year
Pilot Test & Data Collection
4 modules each per grade 6,7,8
Redesign & Finalized
Fall & Spring Design Team Meetings
Capacity Building Data Collection & All Data Analysis

Table 1. CSIMMS Iterative Design Process

Brainstorming:

Teams should identify an important mathematical concept that may be made more accessible to students through the use of some aspect of programming.

Module Design

Once a mathematical topic is identified and the CS approach to be used is identified (i.e., a balloon moving up and down, a sprite drawing a polygon), then the planning focus should shift to a deep emphasis on the mathematics to be learned integrated with principles of mathematics and CS learning.

To maintain **a focus on important mathematical and computer science ideas**, teams should roughly follow the sequence outlined below**.**

1. Identify 2-3 statements of what they want students to *understand* mathematically (versus *do*).

2. Write a first draft of CS activities that stay focused on providing opportunities for students to develop an understanding of those ideas.

   - Following this first draft, the team should examine the Principles of Design to ensure that the overall module and individual tasks adhere to these principles
   - Teams should look for mathematical concepts that require consideration by the entire class in a whole class discussion, as well as identify the kind of reasoning that needs to emerge in this discussion from students. These concepts should be tied to the big ideas of the unit.

3. In a second pass through the activities, teams should look at the Florida math standards and identify a small set of standards that could be met with the module.

   - Following this second draft, the team should examine the Principles of Design to ensure that the overall module and individual tasks adhere to these principles

4. Tasks/Activities should be revised so that those standards are met in a way that can be assessed.

Following this iterative design process, tested content will be addressed, but procedural standards won’t drive the modules.

Table 2. Modules Produced and Refined in Years 1 and 2.

Year 1	
6th Grade	Introduction to Scratch
	Factor Pairs
	Combining Like Expressions: Perimeter of Rectangles and Squares
7th Grade	Introduction to Scratch
	Adding & Subtracting Integers*
	Angles*
8Th Grade	Introduction to Scratch
	Transformations *
	Polygons*
Year 2	
6th Grade	Introduction to Scratch
	Ratio and Proportional Reasoning
	Understanding Means
7th Grade	Introduction to Scratch
	Integer Addition
	Integer Subtraction
	Ratios and Proportional Reasoning
	Drawing Polygons and Exploring Angles
8Th Grade	Introduction to Scratch
	Transformations
	Reflections
	Rotations
	Drawing Polygons:: Sums of Interior and Exterior Angles

*Modules that underwent significant revision in Year 2.

Table 3. Demographics of Schools served by the Mathematics Teachers in the Design Team.

School	Total Enrollment	Race/Ethnicity					FRL Eligibility (%)
		White	Black	Latinx	Asian	Other	
1	1690	49.1	29.1	14.4	3.0	4.3	31.3
2	866	37.8	52.1	4.8	1.5	3.3	52.4
3	469	6.6	87.8	3.6	0.0	0.0	71.2

Table 4. Principle of Design for Computer Science Integrated with Mathematics Modules:
Module and Task Design Questions

Questions to ask of Overall Module Design and Task Sequence

A. Is the module problem-driven (versus being overly guided)? Said, another way, is there an intellectual need to engage in the unit (to either achieve the goal or be more efficient)?

B. Does the task/activity sequence fit with a coherent and concise story line with each day's tasks (e.g. each section) targeting at most one math and/or computer science big idea? Note, this requires attention to the sequencing and building of ideas with regard to both mathematical and computer science ideas.

C. Task sequences should be structured to progressively develop intellectual need for key mathematical ideas across a sequences of tasks in the following way (informed by Realistic Mathematics Education, or RME, instructional design heuristics).

D. Are there regular opportunities to allow for student personalization of code (i.e., changing their sprite or background) as a vehicle to enhance student engagement and can students return to these opportunities regularly?

- These opportunities could appear in the core mathematical/CS tasks following students' day to day work on programming or could be opportunities to work after the tasks are completed.
- Consider building in short or "easy" days early in the unit to allow for all students (not just early finishers) to engage in this customization work—this is particularly important for introductory modules to gain student buy-in for programming in Scratch, but should be a feature for all units.

E. Other considerations for modules are motivated by the modules ability to fit into a teacher's larger curriculum as well as what students can realistically grapple with and learn from each day. These include:

- Are the modules no more than 4-5 days in length? (if so revise)
- Do the modules introduce more than 1 big idea each day (computer science or mathematics)? (if so revise)
- Do the modules have more than 2-3 big, related mathematical ideas per unit? (if so revise)

Questions to ask of Individual Tasks

F. Does each task/activity provide opportunities for students to develop an important mathematical idea or ideas? An important computer science idea?

G. If a task/activity asks students to generalize (and/or represent) a relationship, do students have multiple opportunities to explore the relationship with specific numerical values BEFORE they are asked to generalize?

H. Does the task give students opportunities to explore/develop/test their own code before trying to read/interpret code?

I. Does the task have multiple entry points (different ways for students to enter the problem)? For example:

- Does it ask students to predict what might happen first?
- Is it goal driven (is there a clear goal for students to accomplish)?
- Can it build on a range of student understanding and/or ability level?

J. Does the task have multiple correct and/or productive solution pathways? For example:

- Does it allow for a range of student thinking and reasoning?
- Can students test to find out if their idea (prediction) works?

K. Do the requirements for student code include giving immediate feedback on mathematical thinking (i.e., correct or not) to the students?
L. Do the student tasks statement give students the opportunity to personalize code as a part of the curriculum (i.e., have fun, play around with programming, etc.)?