Assessing college-ready computational thinking

DRK-12 PI Meeting
June 2021

PIs: Mark Wilson, Karen Draney, Richard Brown
Yukie Toyama, Rebecca McNeil, Sean Tan, Richard Patz

Graduate School of Education, University of California, Berkeley
National Math and Science Initiative (NMSI)

This project is funded by the National Science Foundation, grant # 2010314. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Background

- UC Berkeley BEAR Center and NMSI have been collaborating to develop assessments focused on critical reasoning for college readiness in two domains:

  (a) problem-solving using mathematics, and

  (b) data-based decision making.

  Now, we are developing a third domain on computational thinking.
- Computational thinking is a ***fundamental analytical ability*** for all students, joining the ranks of reading, writing, and arithmetic (Wing, 2006)
- Computational thinking applies not only to computer science, but also to a variety of STEM disciplines, as well as the arts, humanities, and social sciences (Bundy, 2007; NRC, 2010, Perković et al., 2010; Wing, 2008)
- Our project thus seeks to develop valid, reliable, and fair assessments in **college-ready computational thinking (CoT)**
  - Applicable across a wide array of disciplines (not just Computer Science)
  - Not associated with particular curriculum or programming language
  - Supports both formative and summative uses for teachers and students in general-purpose high school classes and introductory college courses

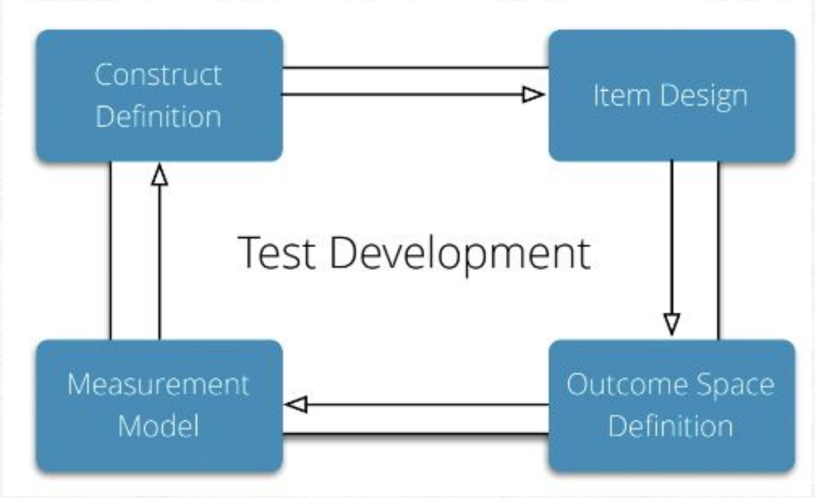
Research questions

- This project currently focuses on two questions:
  1. **What are the most important elements of computational thinking for college readiness?**
  2. **What kinds of assessment items will yield the most usable diagnostic assessment for high school students and teachers with a high degree of reliability, validity, and usability?**
- In addition, we also hope to address the following at a later stage:
  3. How will teachers use this assessment in their classrooms to aid students in improving their computational thinking skills?
  4. Will the use of this assessment in a formative way result in improvement in student performance in end-of-course college-ready tests of mathematics?

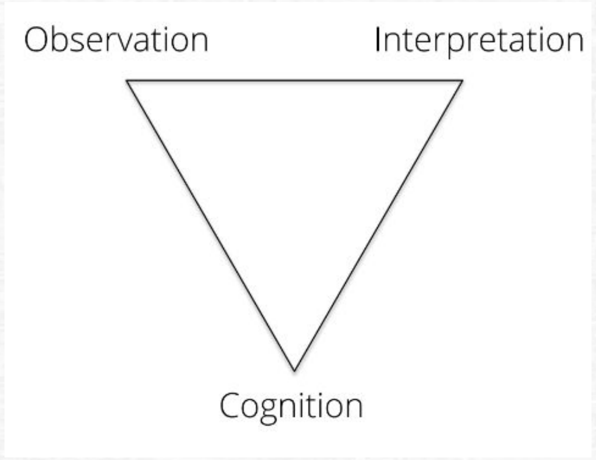
Assessment design

We use a principled assessment design framework known as the BEAR Assessment System (BAS, Wilson, 2005), which includes four building blocks and associated technology tools to be used for constructing quality assessments.

- The building blocks constitute steps in a cycle of development, which may be repeated several times in order to refine various assessment components.
- The four building blocks also embody the three foundations from the NRC Assessment Triangle (NRC, 2001)



BAS with Four Building Blocks



NRC Assessment Triangle

Definition of college-ready computational thinking (CoT)

“Student abilities to design and evaluate algorithmic solutions to real-world problems in substantive domains, often iteratively, in formats that allow both humans and/or computers to implement them.”

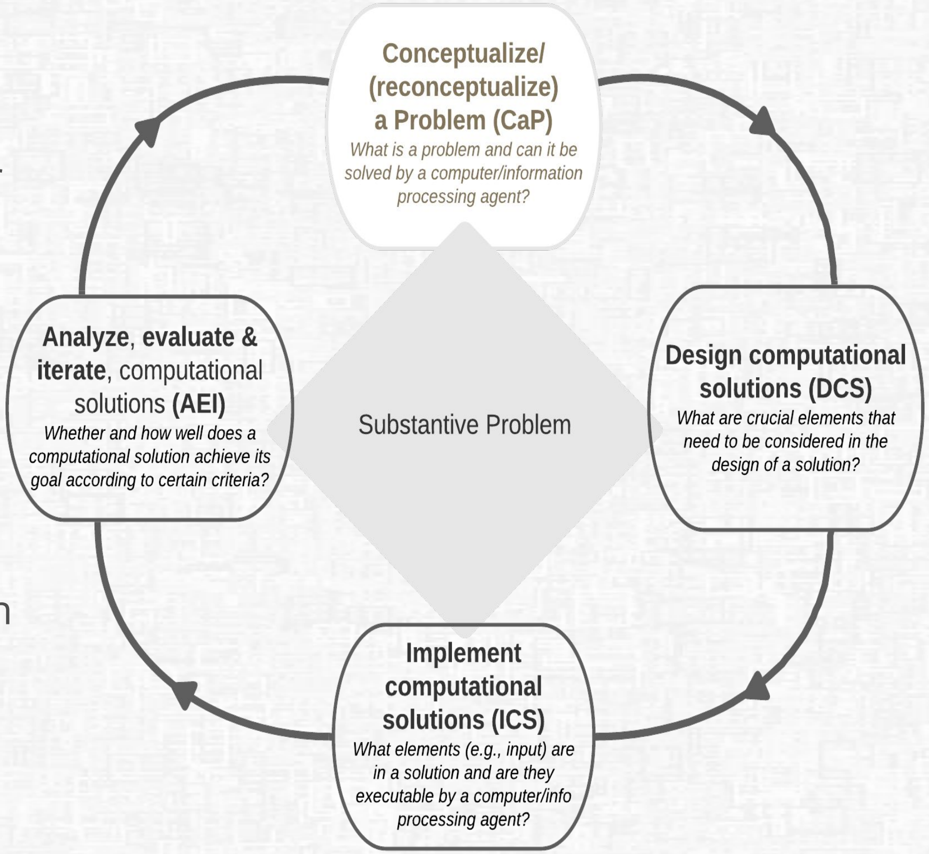
Our current definition is:

- aligned with the view of computational thinking as problem solving (e.g., Aho, 2011; Fraillon et al., 2019; Wing, 2011);
- consistent with Conley (2008)’s view that formulating & solving routine and non-routine problems is a key cognitive strategy for college readiness.
- but is rather narrowly focused on “algorithmic” solutions;
    - we are looking at broadening the definition to include computational practices that go beyond problem solving, such as modeling and simulation, to better understand how systems work (e.g., Weintrop et al., 2016).

CoT assessment framework

The framework comprises four dimensions (or constructs):

- May be viewed as an iterative sequence of steps starting with problem conceptualization and design; however, beginners may start by implementing/modifying existing solutions (Lee et al., 2011)
- Each dimension/construct is composed of an underlying continuum in the form of a construct map (Wilson, 2005; see next slide).
    - Construct maps can be interrelated to form a larger learning progression (Wilson, 2009) for college-ready computational thinking


Conceptualize/ (reconceptualize) a Problem (CaP)
What is a problem and can it be solved by a computer/information processing agent?
Design computational solutions (DCS)
What are crucial elements that need to be considered in the design of a solution?
Implement computational solutions (ICS)
What elements (e.g., input) are in a solution and are they executable by a computer/info processing agent?
Analyze, evaluate & iterate, computational solutions (AEI)
Whether and how well does a computational solution achieve its goal according to certain criteria?
Substantive Problem

*CaP is currently not one of the constructs under development due to the nature of our assessments, but may be considered in the future.

Construct map: Designing computational solutions (DCS)

increasing sophistication

Level	Design
6. Strategic/ Step Beyond	**Fluidly** designs **multiple** solutions with **generalization & creativity**. Articulates trade-offs among solutions/competing goals. **Does not foreclose on known solutions.**
5. Integrated / relational - complex	Designs a **generalizable** solution that can be applied to a range of instances. Beginning to attend to special cases. (Re)frames a problem into a familiar type.
4. Integrated/ relational - simple	Designs a solution(s) to **a problem** with **relational understanding** of multiple subparts with complex operations (e.g., loop, if-then-else). **Beginning to attend to the size/nature of a problem space** and the range of possible solutions.
3. Multi-part solution	Designs **a solution with a linear/discrete sequence of substeps.** May identify a part that can be **automated.**
2. Simple/partial solution	Identifies **a simple/partial solution that may work just in one instance**, or a **general principle/approach** without much specificity.
1. Attempting	Attempts to design with appropriate vocabulary but cannot provide a meaningful response.
0. Not yet evident	No evidence

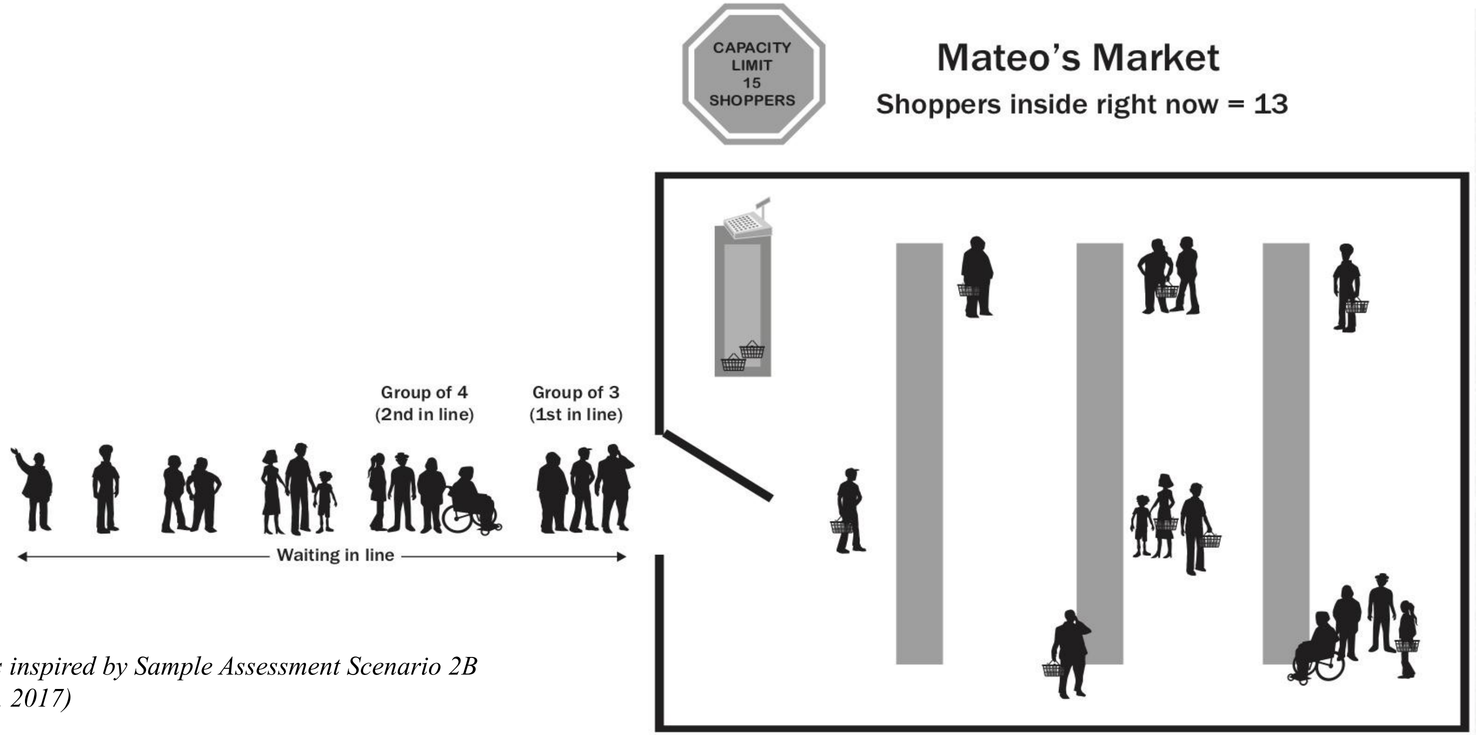
highlighted *= updated as we examined the Fall 2020 pilot data and calibration results*

Sample DCS task: Market

[ABRIDGED VERSION]

Due to social distancing guidelines, a maximum of 15 customers are allowed inside a market at any given time. Customers arrive in groups of varying sizes, and groups are allowed to enter in the order they arrive (but groups may leave the market in any order).

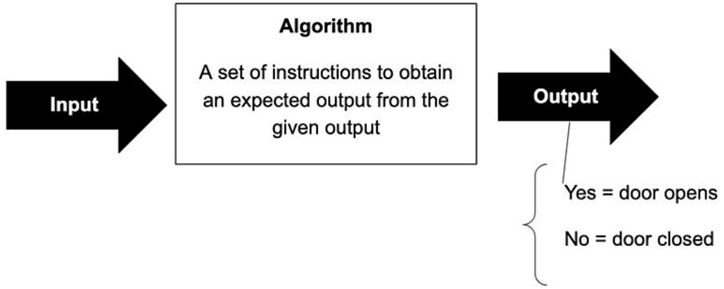
Currently, there are 13 people inside the market, and 6 groups are waiting in line outside as shown below.



Note: This task was inspired by Sample Assessment Scenario 2B (Witherspoon et al., 2017)

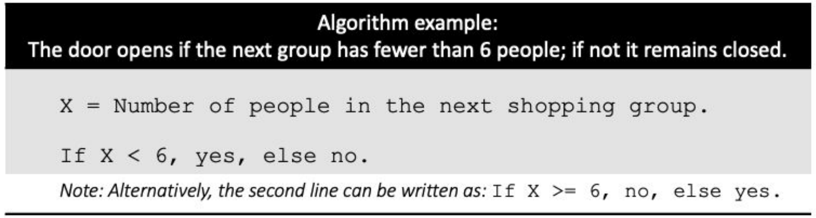
Sample DCS task 1: Market (continued)

The market owner wants you to develop a computer program that automatically opens the entrance door when there is enough capacity in the market to accommodate the next group. For this program to work, you have to specify the input, an algorithm, and the output as outlined below.


Input
Algorithm
A set of instructions to obtain an expected output from the given output
Output
Yes = door opens
No = door closed

[a] What input(s) needs to be provided to the program?

[b] Using the input(s) you identified in [a], write the algorithm that determines whether the door opens (output = "yes") or remains closed (output = "no"). Write your answer following the format of the algorithm example below.



Algorithm example:
The door opens if the next group has fewer than 6 people; if not it remains closed.

```
X = Number of people in the next shopping group.

If X < 6, yes, else no.
```

Note: Alternatively, the second line can be written as: `If X >= 6, no, else yes.`

[c] The market owner does not know much about algorithms but wants to know what your algorithm is telling the computer to do. Write a short explanation for him in plain English.

Scoring guides for Market & Video game

Score	Description
4. Integrated/ relational: simple	● Identifies two variables: *x = # of ppl in the next group*; *y = # of ppl inside*) and the constant: *15 = max # of ppl allowed*, AND ● Designs a complete and correct solution using the three elements: *If x + y <= 15, yes, else, no*.
3. Multi-part solution	● Identifies the 3 elements specified above BUT the solution offered is fragmented / incomplete.
2. Simple/ partial solution	● Identifies only 1-2 elements specified above, OR a specific case for a certain output: e.g., *Door opens if 11 ppl are inside & next group has 4 ppl*, OR ● Identifies a general principle: e.g., *The door opens when there is enough space inside.*
1. Attempting	● Repeats algorithm example given in the prompt: *If x <6, yes, else no.*
0. Not yet evident	IDK, Off-topic.

Project Activities

Literature review

Construct maps & item development

Summer 2020

Cognitive labs

- 8 students
- 32 items

Item panel & review

Internal within research team
External with NMSI coaches

Pilot testing 1

- 208 students enrolled in an AP CS course
- 2 forms, 37 items

Fall 2020

Scoring, calibration & analysis

Advisory panel meeting

Item revisions

Winter 2020-21

Pilot testing 2

- 120 students enrolled in an AP CS course
- 2 forms, 38 items

Spring 2021

Further analyses standards setting

Summer 2021

Constructs & items refinement

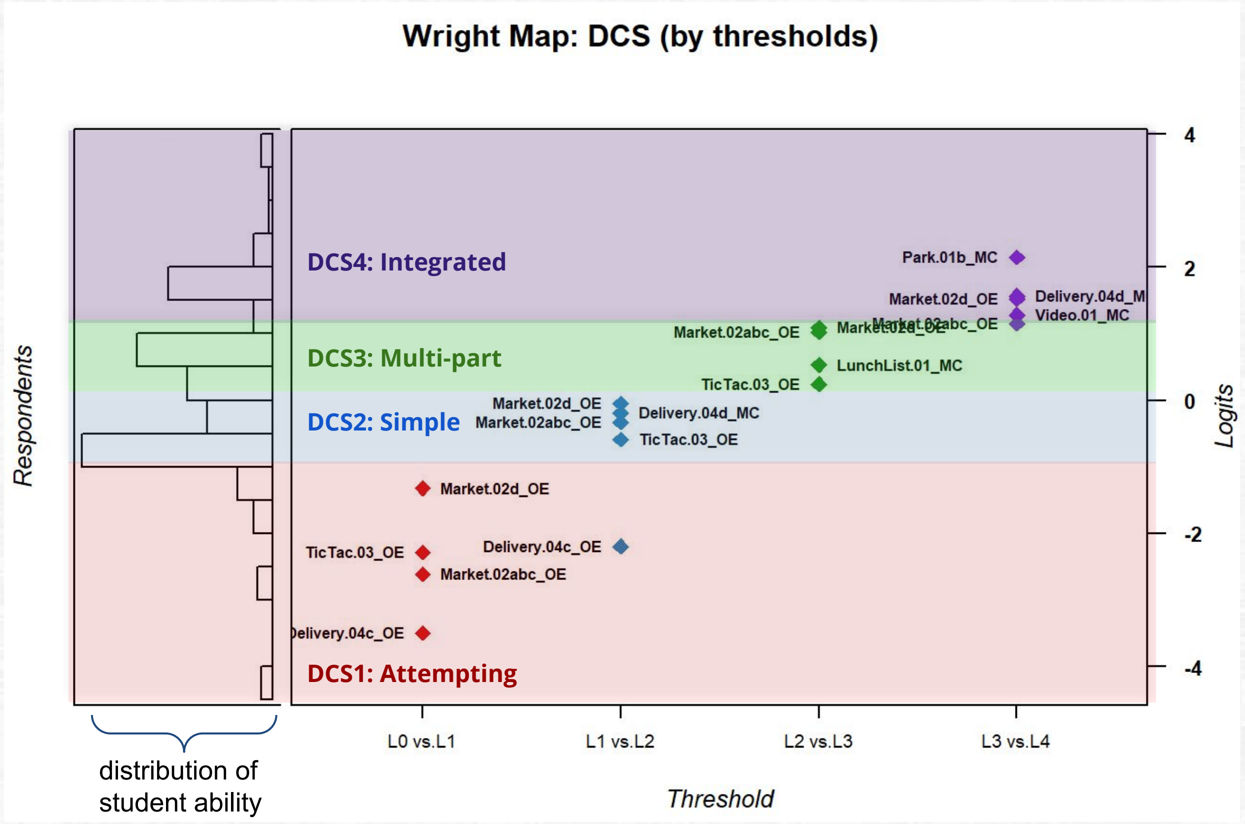
Item generation

Preliminary findings

Based on the Fall 2020 Pilot

Analyses for the other two constructs are underway

Construct validity (WrightMap)

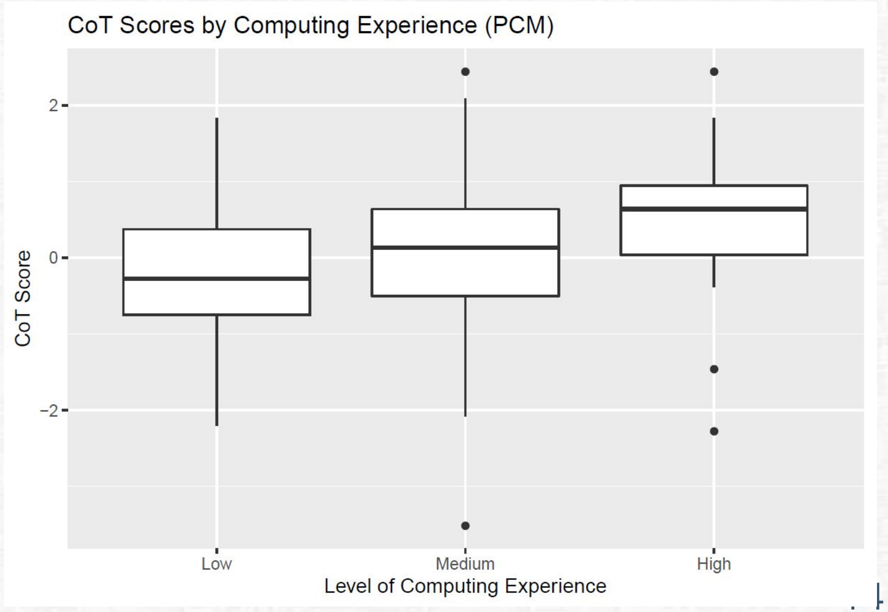


- Each construct was calibrated separately, using Master's (1982) Partial Credit Model
- The WrightMap shows **an upward trend in threshold difficulties** & **banding** as construct levels increase, providing **validity evidence** for our hypothesized structure of the Design construct.
- As we analyzed the data, we **iterated** on our construct level definition, scoring guides, and item design.

Other validity and reliability evidence

- Overall CoT EAP/PV reliability: 0.87
- Response process: Students tend to spend more time and exhibit/report greater difficulty when solving higher level tasks during cognitive labs.

- Relationship to prior experience
  - Students with high level of computing experience (e.g., coding, robotics, web design) tended to have higher CoT scores.



What we learned…

- Preliminary results offer empirical support for our hypothesized construct levels for the DCS construct:
- Iterative assessment development is crucial
    - In addition to item paneling, multiple rounds of cognitive labs as well as iterating on items, scoring guides, and construct level definitions against empirical results (WrightMap) are key in refining the assessment *and* our understanding of the constructs.
- Balancing the authenticity of a problem and the amount of subject-matter knowledge required is a challenge in designing CoT assessment tasks.

Future steps

- Continue assessment development cycles:
  - Refine constructs, especially how to differentiate them
  - Refine construct levels, items, and scoring guides
  - Develop more items with dynamic features
  - Collect further validity and reliability evidence

    e.g., relationship with other variables including teacher ratings of students’ CoT proficiency and AP CS exam results
- An interview/survey study with college faculty
  - to explore what they would like students to know and be able to do upon enrolling into their introductory courses
- Usability and implementation studies
  - to learn about students and teachers’ uses and their perspectives

References

Aho, A. V. (2011). Computation and computational thinking. *ACM Ubiquity, 1*, 1-8.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, *1*, 6769.

College Board. (2017). *AP Computer Science Principles Course and Exam Description: Including the Curriculum Framework*. New York: The College Board.

Fraillon, J., Ainley, J., Schulz, W., Duckworth, D., & Friedman, T. (2019). *IEA International Computer and Information Literacy Study 2018 assessment framework*. Cham, Switzerland: Springer.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32-37.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, *47*, 149-174.

National Research Council (NRC). (2001). *Knowing what students know: The science and design of educational assessment*. Washington, DC: National Academy Press.

National Research Council (NRC). (2010). *Committee for the workshops on computational thinking: Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press.

Perković, L., Settle, A., Hwang, S., & Jones, J. (2010). A framework for computational thinking across the curriculum. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 123-127).

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147.

Wilson, M. (2005). *Constructing measures*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Wilson, M. (2009). Measuring progressions: Assessment structures underlying a learning progression. *Journal of Research in Science Teaching*, *46*(6), 716-730.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society*, *366*(1881), 3717-3725.

Wing, J. (2011). Research notebook: Computational thinking—what and why. *The Link Magazine*, *6*.

Witherspoon, E. B., Higashi, R. M., Schunn C. D., Baehr, B., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Trans. Comput. Educ.*, *18*, 1.

Thank you

NMSI

Ryan Higgins
Melissa Estremera
Diane Keller
Paula McKinney
Erica Roberts
Marilyn Turmelle
Participating teachers & students

BEAR Researchers

Jerred Jolin
James Mason
Perman Gochyyev
Xingyao (Doria) Xiao

BEAR IT

Ana Maria Albornoz Reitze
David Torres Irribarra

Advisory Panel

Howard Everson
Carolyn Huie Hofstetter
Pedrito Maynard-Zhang
Zachary Pardos
Roy Pea
Kathleen Scalise
Finbarr (Barry) C. Sloane
Michelle Wilkerson

Graphic designer

Greg Klinger

Evaluator

Carolyn Huie Hofstetter