International Society of the Learning Sciences

Broadening Participation in STEM-based Computational Modeling by Leveraging Alternatives to Programming

Kevin W. McElhaney, Digital Promise, kmcelhaney@digitalpromise.org
Satabdi Basu, SRI International, satabdi.basu@sri.com
Nonye Alozie, SRI International, maggie.alozie@sri.com
Nicole M. Hutchins, University of Florida, nicole.hutchins@ufl.edu
Arif Rachmatullah, SRI International, arif.rachmatullah@sri.com
Kelly Mills, Digital Promise, kmills@digitalpromise.org
Gautam Biswas, Vanderbilt University, gautam.biswas@vanderbilt.edu

Abstract: Computational models (CMs) offer pre-college students opportunities to integrate STEM disciplines with computational thinking in ways that reflect authentic STEM practice. However, not all STEM teachers and students are prepared to teach or learn programming skills required to construct CMs. To broaden participation in computing, we propose instructional approaches that integrate STEM with CMs without requiring students to program, thereby alleviating challenges associated with learning how to program.

Introduction and rationale

Computational models (CMs) offer promise for broadening participation in computing by integrating computational thinking (CT) with STEM disciplines (Sengupta et al., 2013) in ways that are authentic to professional STEM practice. A CM is a mathematical representation of a system created using a formal notation (e.g., programming language), enabling system behavior to be studied using computer simulations. However, developing CMs as part of STEM instruction may place unreasonable demands on students and teachers, such as adding instructional time for students to learn programming, adding professional development requirements for teachers, and dividing students' attention between learning programming and STEM concepts.

We propose two approaches to engaging students and teachers in STEM learning using CMs in ways that do not require programming, as programming need not necessarily be a focal learning goal in order for learners to engage in CT practices. Engaging students in *interpreting and evaluating CMs* or *exploring and evaluating simulations* leverages the affordances of decoding and critique (Wagh et al., 2017) and maintains an authentic focus on CT in ways that may better address some students' and teachers' specific needs.

Framework for integrating STEM and computational modeling

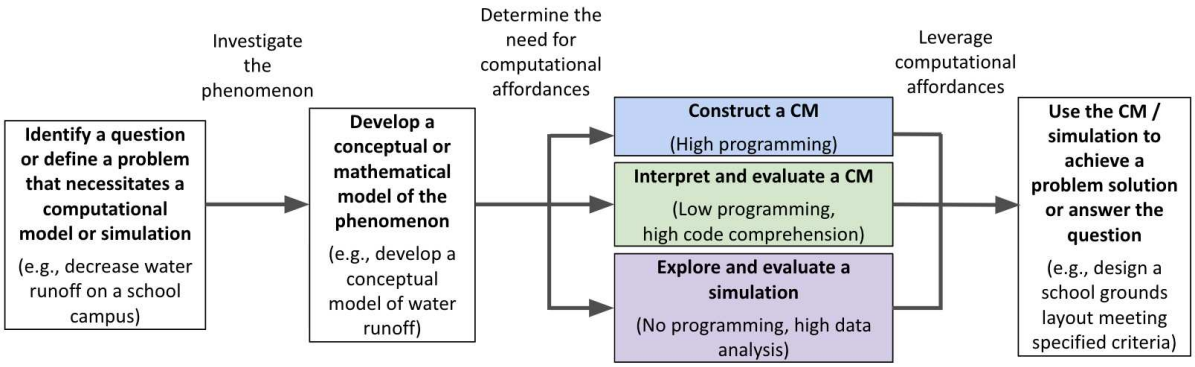
We adapt a framework for integrating science, engineering, and CM construction (McElhaney et al., 2020) to encompass learning opportunities that integrate CM with STEM phenomena more generally (Figure 1). First, learners identify a question or define a problem whose answer or solution necessitates a CM. This question or problem could be in an engineering, mathematical, or social science context. Second, learners investigate the phenomenon in order to develop a conceptual or mathematical model of the system behavior. This conceptual model must be able to be subsequently represented as a CM or simulation. The model also creates a need for computational affordances (e.g., automation) that address the question or problem. Third, learners engage in CM activities that are appropriate for their level of CT knowledge, experience, and learning context. We elaborate on approaches to designing CM activities below. Finally, learners use the CM or simulation to answer the overarching question or solve the problem, such as by generating a design, explanation, or prediction.

CM activities typically ask students to construct a CM incrementally (Figure 1, blue box), engaging them in the CT practices of programming and debugging. Constructing a CM is not required for students to engage in other CT practices (Weintrop et al., 2016) such as recognizing computational problems, using CMs, assessing CMs, or analyzing data from CMs. Instead of constructing the CM, other approaches can ask students to assume the role of someone who does not code, but rather works adjacent to the coder, such as a scientist, economist, or policy official. In this role, the student can be given several versions of a CM and be asked to determine which one exhibits the desired behavior. New code or simulation features can be added incrementally and evaluated in turn, in the same sequence as students would decompose and construct the CM itself.

The *interpret and evaluate a CM* approach (Figure 1, green box) places less emphasis on programming and more emphasis on code comprehension, relative to constructing a CM. Students may be provided with a simulation and its underlying code, which may be incomplete or buggy. Curriculum activities engage students in comprehending the code, predicting its output for different inputs, determining what parts of the code are missing

or incorrect, and suggesting what revisions to the code are needed in order to produce the desired model behavior. Students will need to engage in activities designed to promote learning of programming concepts.

Figure 1
Framework for integrating computational modeling into STEM instruction.



The *explore and evaluate a simulation* approach (Figure 1, purple box) does not engage students in any programming (because they cannot see the code), but places high emphasis on computational practices related to data analysis. Students are provided with incremental versions of the simulation that exhibit only some parts of the desired behavior, or have bugs in the underlying code. Students must analyze data generated by running simulations in order to identify which behaviors work and which do not. To engage in this approach, students must have a method of tracking input parameters used to run the simulation and their corresponding outputs.

Discussion and implications

Our framework offers an alternative to established instructional sequences (e.g., Use-Modify-Create, Lee et al., 2011), that intend to scaffold stand-alone programming activities. While such sequences are appropriate for supporting general program creation, they may not align with the goals of STEM-based CMs, where using the CM is often the end goal of the modeling process. Our proposed alternatives to programming reflect the activities of STEM professionals who use, assess, and modify CMs and simulations to understand scientific or social phenomena, but do not develop CMs themselves. Our ongoing research examines the relative affordances of these three CM-based approaches for integrated STEM+CT teaching and learning. Research-based insights about how these approaches to CM-based instruction inform STEM teaching and learning for diverse students and teachers are needed to improve implementation of STEM+CT instructional materials and teacher supports.

References

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.

McElhaney, K.W., Zhang, N., Basu, S., McBride, E., Biswas, G., & Chiu, J.L. (2020). Using Computational Modeling to Integrate Science and Engineering Curricular Activities. In M. Gresalfi & I.S. Horn (Eds.). *Proceedings of the 14th International Conference of the Learning Sciences (ICLS) 2020*, Volume 3 (pp. 1357-1364). International Society of the Learning Sciences: Nashville, TN.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*, 351-380.

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*, *54*(5), 615-641.

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*, 127-147.

Acknowledgments

We acknowledge the support of National Science Foundation grants DRL #2055609 and #2055597. We are grateful for the suggestions of three anonymous reviewers.