Book Notes

Voicing Code in STEM: A Dialogical Imagination
by Pratim Sengupta, Amanda Dickes, and Amy Voss Farris
Cambridge, MA: MIT Press, 2021. 232 pp. $28.00 (paper).

The field of preK–12 computing education has, in recent years, often focused on teaching computational thinking, which "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). In short, there are specific habits of mind that are acquired when learning to program computers, and these habits of mind can benefit nonprogrammers as well (Denning, 2017). Some scholars have also argued that computational thinking as a framework is insufficient for understanding what learning to program entails and that, instead, we should be focused on teaching and studying computational making (Rode et al., 2015), computational participation (Kafai & Burke, 2013), or computational action (Tissenbaum, Sheldon, & Abelson, 2019). While some might consider this to be a purely academic argument, each phrase attends to different values, which in turn have profound implications for curriculum developers, teachers, and students. For example, this debate of *what* to teach in computing education has led to discussion of *how* to teach computing, and approaches to teaching computing can take a fairly technocentric view, with a focus on the technology itself without consideration of contextual elements within the learning environment (Brennan, 2015). This can look like focusing on building new programming environments as a way to improve learners' experiences rather than taking a closer look at teachers' professional learning needs and working conditions or the design of the accompanying curriculum for these programming environments.

In *Voicing Code in STEM: A Dialogical Imagination*, Pratim Sengupta, Amanda Dickes, and Amy Voss Farris advocate for the need to push past technocentrism to reframe coding. They take a critical phenomenological turn to consider dialogic lenses and argue that we should "shift from viewing coding as production of computational artifacts to voicing computational utterances" (p. 24). In other words, instead of focusing on the learner, artifacts the learner has made, or the technology the learner used to create the artifact, the authors ask us to explore the conversational exchanges learners make with one another as well as with their teacher.

In the first chapter, the authors draw from Mikhail Bakhtin's work on language and dialogue (primarily from *The Dialogic Imagination*) to note that they

"position computational utterances as elemental pieces of experience that are the sites at which the constancy, historicity, and systematicity enter into contact and struggle with unique, situated performance" (p. 26). This push toward a focus on utterances creates opportunities for inclusivity and computational heterogeneity, which can value multiple ways of learning, speaking about, and creating artifacts with code. Chapter 2 expands on their theoretical framework of Bakhtinian dialogical imagination in the context of code, while the following five chapters offer considerations of different aspects of their theory through examples drawn from different studies conducted by the authors in K–12 science, technology, engineering, and math contexts.

Specifically, the authors focus on data collected from classrooms where students and teachers are interacting with LOGO-derived programming languages designed as introductory programming languages for children. In chapter 3, for example, they explore the experiences of two fifth-grade students learning to program a turtle to move on the screen through in-depth analysis of twenty-three minutes of collaboration with each other, examining the students' perspectival shifts to highlight how a focus on students' conversation, instead of the code they have worked on or the devices they have engaged with, can illuminate their learning processes. The final chapter offers a "radical reflection" (p. 191) on computational heterogeneity, ending with three lessons for avoiding technocentrism.

Across *Voicing Code in STEM*, the relationship between the empirical work and the theoretical contribution is clear and compelling, and the authors offer a window into moments of student learning by sharing images, transcript snippets, and samples of student code. The book is well written, and the empirics are legible even for a generalist reader. Each of the main chapters offers helpful ways of thinking about code as voice, and even in these brief moments students' voices are clearly heard. The focus on computational utterances foregrounds dialogue between students and their teachers, centering the people who are most deeply impacted by this work. The deep and thoughtful attention to the rich micro-interactions of learners creates opportunities for scholars to consider the pedagogical affordances of the learning environment, such as how the relationships among students and between students and teachers can inform and support their learning. I found myself reminded of the power of talking to students and learning more about their experiences. In chapter 7 we hear about the computational models that Shenice, a fifth grader, builds. While the models themselves appear to be fairly straightforward (a cross, drawn and rotated multiple times), it is through the rich description of Shenice's learning experience that we learn about her close relationship to her local church, the authors noting that this phenomenological account "certainly can include analysis of her computational work, but should never be subsumed by it" (p. 179). I hope that these rich examples of attending to what students say as they work can inspire other researchers, educators, and learn-

ing designers to attend not only to learners' artifacts but also to their experiences, particularly within their sociocultural contexts.

But while the empirics are clear, the concluding "radical reflection" could have gone further in considering not only how computer code can be considered voice in STEM contexts but what it might mean for other disciplines. In the fields of cultural production and youth development, scholars have long highlighted youth voice in digital media production (Kafai & Peppler, 2011). How might we bridge the gap between understandings of cultural production in the arts and humanities with STEM?

In addition, this argument is explicit in privileging more ephemeral forms of communication, such as utterances, over "device-level engagement." While technocentric approaches often focus too much on the technology, or the devices, we might consider differences between paying attention to devices as opposed to the artifacts that learners are able to craft with the devices and the programming languages that they learn to use. I also wonder what the implications are for educators around what they should attend to in the classroom. And even as we think about the dialogical imagination, how do power and privilege impact whose voices are heard more?

Voicing Code in STEM offers radical provocations that shifted my own thinking about what is important and meaningful as learners deepen their understandings of STEM concepts and practices, and I am excited to see how this is taken up by other researchers and educators, as well as how we can continue to center learners and what they produce in future work.

PAULINA HADUONG

References

Brennan, K. (2015). Beyond technocentrism: Supporting constructionism in the classroom. *Constructivist Foundations*, *10*(3), 289–296. Retrieved from http://constructivist.info/10/3/289

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. doi:10.1145/2998438

Kafai, Y. B., & Burke, Q. (2013). *The social turn in K–12 programming: Moving from computational thinking to computational participation.* Proceedings of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO. doi:10.1145/ 96.2445373

Kafai, Y. B., & Peppler, K. A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. *Review of Research in Education*, *35*(1), 89–119. doi:10.3102/0091732X10383211

Rode, J. A., Weibert, A., Marshall, A., Aal, K., von Rekowski, T., El Mimouni, H., & Booker, J. (2015). *From computational thinking to computational making.* Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan. doi:10.1145/2750858.2804261

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM*, *62*(3), 34–36. doi:10.1145/3265747

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. doi:10.1145/1118178.1118215