International Society of the Learning Sciences

Possibility, Perspectives, Personal Investment, and Process: How K–12 Teachers Support Programming Projects Through Assessment

Paulina Haduong, Karen Brennan
haduong@g.harvard.edu, karen_brennan@gse.harvard.edu
Harvard University

Abstract: Creating personally authentic projects can support student engagement and motivation in learning to program. Assessment can play a role in supporting student learning, but it can be challenging for teachers to employ assessment practices that benefit students and their programming projects. In this interview study conducted with 80 U.S. K–12 computing teachers, we explored how teachers supported students' programming projects through their assessment practices. Teachers navigated tensions between their pedagogical aspirations (e.g., supporting creative possibilities) and practical considerations (e.g., logistical constraints) as they supported students projects by (1) encouraging possibility through rubrics; (2) inviting multiple perspectives through feedback; (3) increasing personal investment through self-assessment and resubmission; and (4) surfacing process through documentation and conversation. Findings from this work support understandings of how assessments can support or constrain learning through creating programming projects.

Introduction

Learning to program through the creation of personally authentic projects can be an inclusive and equitable pathway into computing, supporting learner motivation through experiences that are "personally or culturally meaningful in the mind of the learner" (National Academies of Sciences, Engineering, and Medicine, 2021, p. 30). In elementary school teacher Oscar's classroom, students created Scratch projects about their favorite locations in their neighborhoods, and in middle school teacher Tara's classroom, students designed 3D models of objects for their family members. Assessment is a critical component of supporting student learning through creating projects (Earl, 2012), but programming projects can be challenging to assess (Brennan & Resnick, 2012; Shute et al., 2017). Without clearer understandings of how to support students' programming projects through assessment practices, teachers may be deterred from designing opportunities for students to learn through the creation of programming projects altogether (Brennan, 2015). In this study, we investigated: *How do K–12 computing teachers support students' programming projects through their assessment practices?*

Background

When students engage in creating projects, well-designed and ongoing assessment is critical for supporting student learning (Barron & Darling-Hammond, 2010). It can be difficult to design effective assessments, however, because assessing and understanding student projects can be a complex process (Brinkerhoff & Glazewski, 2004; Pedersen & Hobye, 2020). Because students' projects and project processes necessarily diverge from one another, teachers often draw on pedagogical content knowledge (Jones & Moreland, 2005) and assessment literacy (Lew & Nelson, 2016) to effectively support many students simultaneously. A lack of time—not only for students to make projects but also for teachers to assess those projects—is a common challenge (Kleinert et al., 1999; Lumadi, 2013). Moreover, assessment practices themselves run the risk of stifling student motivation and interest (Earl, 2012). As Barnes et al. (2015) write, "assessment practices hold power that can enhance learning and democratic practice or can be used to punish and control learners, teachers, and schools" (p. 298).

In the context of teaching and learning computer programming, teachers may face unique challenges when supporting students in engaging in projects. While students' programming projects are easily visible, student learning may be harder to understand and support (Brennan & Resnick, 2012). In computing education, manual checking of student code is a customary and time-consuming form of assessment (Grover, 2017), though some researchers and educators have developed automated assessment tools (Blikstein et al., 2014; Douce et al., 2005; Ihantola et al., 2010). Even with automated tools, teachers need a high degree of pedagogical content knowledge to assess student learning through programming projects, particularly when students have been encouraged to take their projects in directions that feel personally authentic to them. Students' programming projects can vary, even when they incorporate the same or similar computational concept (Capraro & Slough, 2013). Students might engage with different computational concepts, practices, and/or perspectives by attempting projects in different

ways (Brennan & Resnick, 2012). In the U.S., although there has been increased demand for and investment in computing education for students, there has been a lack of commensurate support for teachers' professional learning (Blikstein, 2018). Limited access to professional learning can stifle computing teachers' development of pedagogical content knowledge and assessment literacy, leading to difficulties in knowing what to teach and how to assess (Bower et al., 2017; Yadav & Berges, 2019). Without the requisite assessment literacy, computing teachers sometimes forego the assessment of projects altogether, relying on separate, more "traditional" assessments (i.e., handwritten exams) as ways of understanding student learning. These exams typically focus on student acontextual conceptual knowledge, and a lack of assessment around students' projects can unintentionally minimize how much students value their own projects (Yadav et al., 2015).

Method

In this study, we investigated the assessment practices that teachers implemented in their classrooms. We explored how teachers' aspirations for supporting students' programming projects informed their assessment practices, as well as how teachers negotiated challenges in employing these assessment practices.

Teacher participants and interview design

Eighty U.S. K–12 computing teachers were recruited via social networks (e.g., Twitter) and email listservs (e.g., Computer Science Teachers Association). Half of participating teachers worked with grades 9–12 and half worked with grades K–8. 20 teachers taught at least one Advanced Placement (AP) course. While some had prior CS experience, most participants taught other subjects (e.g., math, science, or the arts) and had begun teaching computing because of personal interest or school demands. They taught in rural, suburban, and urban districts in 27 states. School contexts varied, with two-thirds of teachers working at public schools. Some K–5 teachers taught all subjects, including computing, to their own class of students, while others taught computing to all students in the school in their roles as librarians/computer lab teachers.

Individual members of the research team conducted interviews with pairs of teachers about assessing programming projects in computing classrooms. Asking teachers to share and discuss two assessments they used in the classroom enabled us to ground interviews in student work and teacher-generated assessments. The pair design offered opportunities for peer learning and richer data (Haduong & Brennan, 2020). Interviews were virtual, over video chat, and teachers were paired randomly by scheduling availability. Interviews were structured as a three-part turn-based protocol: after the research team member asked a question, each participant responded to the same question. First, we asked participants to define key terms (e.g., "assessment") and how these constructs manifested in their classrooms. Then, participants shared their assessments (e.g., rubrics and student reflective writing), describing how and why these assessments were developed. Finally, we asked participants to share advice or questions they had for other teachers. Interviews averaged 75 minutes, and they were audio-recorded and transcribed. All teachers were assigned pseudonyms, which is how they are referenced in this article. After the interview, each participant received an Amazon gift card ($100).

Data analysis and limitations

Interviews were analyzed through a collaborative and iterative process of thematic analysis (Braun & Clarke, 2012). In the initial rounds of open coding, the first author and another research team member focused on assessment practices (e.g., "feedback"), coding together to develop a preliminary codebook in NVivo, then piloting the codebook by individually coding the same interviews and meeting weekly. Once the assessment practices had been established, the team then focused on developing codes related to each practice's underlying rationale (i.e., aspiration for supporting programming projects), as well as the challenges related to each practice. After a codebook was established, they engaged in split coding, discussing discrepancies and concerns weekly with the second author to arrive at consensus (Cornish et al., 2013). Through reflective memos and regular discussions, the research team worked together to reorganize codes and understand the relationships between aspirations for students' projects, assessment practices, and practice-related challenges. In the final phase, we looked for "discrepant evidence" that was at odds with the existing codebook (Maxwell, 2010) and revised accordingly (see Table 1).

There are several limitations to our approach. In this study, we recruited teachers interested in supporting and assessing student creative programming projects; this call may have excluded teachers who were implementing programming projects in the classroom but didn't feel that their work was "creative." As an interview study conducted remotely, we were not able to observe participants in their classrooms, though we triangulated participant responses with submitted assessment examples. We sampled a broad and diverse group of teachers working in K–12 computing education across the U.S. and found that key practices spanned contexts.

Table 1
Assessment Practice Codes by Teacher (n=80)

Aspiration	Practice	Example	Teachers
Possibility	Rubrics	"I want four sprites, but I don't specifically say what the four sprites are, and that gives them creativity within my constraints."	70
Perspectives	Feedback	"Bringing another person into the conversation helps them push their thinking along because they're testing their ideas out on an actual user … it's really great to get feedback from a peer, and it's great to watch someone play your game."	65
Personal investment	Self-assessment	"Did you do more than you ever thought you could do? And do you keep wanting to do it? That's when [you] can self-assess and say, "I'm starting to get there.'"	53
	Resubmission	"They can keep resubmitting, [and] manage and design their own test loop … they can iterate very quickly."	21
Process	Documentation	"It's like they're walking around with a tape recorder … And they're showing me all of the process that was involved, and then the product ends up only being a small part of it."	29
	Talking to students	"I walk around and they run [their programs] for me, and then I may ask them some questions about, "What did you do? How did this work? Show me what you did here."	68

Findings

In learning environments where students were engaging in programming projects, teachers aspired to encourage expansive possibilities, invite multiple perspectives on student work, increase students' personal investment in their projects, and surface students' process. One way these aspirations informed how teachers designed opportunities for creating programming projects was through their assessment practices, which enabled teachers to differentiate project support based on students' varied needs, skills, and lived experiences. At the same time, teachers negotiated external accountability demands, expectations of what learning and assessment "should" look like, and the time constraints of K–12 class schedules. Participating teachers described supporting programming projects through four assessment practices: (1) encouraging possibility through rubrics; (2) inviting multiple perspectives through feedback; (3) increasing personal investment through self-assessment and resubmission; and (4) surfacing process through documentation and conversation.

Encouraging possibility through rubrics

Having a sense of what is possible facilitates the ability to create personally authentic programming projects. Teachers described how students, unused to opportunities to engage in this type of open-ended project work, struggled to make sense of what they were "supposed" to create. In response to this challenge, most participating teachers (n=70) used rubrics (or scoring guides) when facilitating programming projects, which helped students imagine expansive creative possibilities while understanding expectations for project requirements. Defining expectations helped teachers justify grades. But teachers also felt that too many expectations could limit student freedom. More than half of teachers (n=48) asked students to make projects that included specific computational concepts (e.g., loops or variables) while leaving the type of project and specific implementation up to them, striving to support a diversity of outcomes. A few teachers (n=4) designed rubrics that were purposefully minimal and open-ended (e.g., "Creativity: Student explored and expressed multiple ideas in a unique way") while others (n=12) strove to make them as specific as possible (e.g., "Creativity: Student selected a stage, sprite, and action commands that create an engaging dance performance for the viewers"). Joshua, an elementary school teacher, described how the rubric–as–checklist helped students decide how much challenge they wanted to take on in their projects, while also helping students see what else they might do in their projects.

Several high school teachers noted that they used rubrics despite their limitations, particularly around student motivation. Derek said, "Many students see the rubric as the maximum" while others "see it as the minimum," and Shannon expressed a concern that using rubrics "really lowers the level of creativity and engagement in a project." Teachers also differed in *when* they shared rubrics with students in the project process. Some teachers (n=13) gave rubrics to students at the beginning of the process to help students imagine what was possible. Kayla, a high school teacher, said, "I do give [students] that base rubric so they have a starting point. I find that if I don't, they feel very lost." Some teachers (n=14) used the rubric as a checklist throughout the project

process, though others felt strongly about sharing the rubric only at the end. Damien said, "I don't want them going into checklist mode" in the beginning, because he wanted students to be "exploring and figuring out and not limiting themselves based on the rubric yet." Even though the use of rubrics was a common practice, several teachers raised questions about how rubrics could better support opportunities for students to understand what they might explore within their programming projects.

Inviting multiple perspectives through feedback

Programming projects can benefit from students' access to multiple perspectives, which can spark new ideas and support students when they get stuck. Assessment in support of learning is most effective when delivered in a timely manner (Brookhart, 2008), but teachers noted that, particularly with large classes, assessing student programming work could be burdensome. Erin's biggest problem was "timeliness in getting them feedback … I can't be a bottleneck for anything; kids have to know where they are as soon as they can." Throughout projects, many teachers (n=65) created opportunities for timely and individualized feedback by inviting multiple perspectives on students' projects. Erin, a high school teacher, incorporated peer feedback through "round robin presentations," splitting her class in half and asking each student to present individually to another student for ten minutes before rotating, so that every student was able to share and receive feedback multiple times. Teachers also reported that students valued peer feedback more than teacher feedback; Tracy, a middle school teacher, said, "the most meaningful thing [for kids] is the feedback they get from their peers."

Practicing assessing others' work can inform students' awareness of other solutions and scaffold their ability to assess their own programming projects (Smith et al., 2012). Having other people test their projects was often helpful for students' progress, and Leslie, a K–5 teacher noted that students were "really quick about finding mistakes" in others' work. Peer feedback, however, can take *more*, instead of less, time. Anita, an elementary school teacher, reported that written peer feedback resulted in more work because "it takes a long time for me to read all [the feedback] for the students." Students may also need to learn how to give each other feedback (Beach & Friedrich, 2006). Quentin, a high school teacher, printed and displayed students' code on large sheets of paper, so students could "appreciate other people's code and look at a variety of directions that students took to solve a problem." He then gave students sticky notes and discussed how to provide feedback. To his students, he said, "If I came to your desk and told you that your code was good or bad and didn't say anything else, you'd hate me, right? I'd be a bad teacher. So, be a good peer and provide constructive but respectful feedback."

Feedback could also come from other authentic audiences. In Jill's AP CS A classroom, industry volunteers offered technical expertise that she did not have. Jill, a novice programmer, described how having less content knowledge slowed down her ability to support student work. Because she was not able to understand students' code by reading it, she often had "to test it out because we don't know all the ways the kids' minds work … they're all over the place." A few teachers (n=8) also facilitated opportunities for students to receive feedback from audiences beyond their classroom, such as asking family members, friends, or industry volunteers to play the games they had designed. For students, receiving feedback from others at multiple points during their projects helped them refine their ideas and iterate on their projects.

Increasing personal investment through self-assessment and resubmission

When students are working on personally authentic projects, they are more likely to be invested in their creations. Personal investment can support persistence in the face of challenges, possibly leading to future participation in computing. Assessments can support personal investment, but assessments are often thought of as something done *to* students, instead of *with* students (Earl, 2012). Many teachers (n=60) spoke about tensions between how they wanted assessments to support student learning and the expectations that students, parents, and/or administrators brought to assessments. Teachers wanted to prioritize formative assessments, but the formal structures in which they were operating often demanded summative assessments that felt antithetical to learning goals. Teachers (n=35) also noted uneasy relationships between assessment, motivation, and teacher feedback. In class, high school teacher Lorena tried to encourage students working on projects to "forget about the grade, forget about what I want from you, because you're not doing this for me." But students would push back, expecting "traditional assessments," such as multiple-choice quizzes. Other teachers spoke about students' tendency to focus on grades, rather than reading and incorporating feedback. Ashley, a high school teacher, said, "Most kids honestly just look at the grade, and that's all they want to see," specifying that even though she wanted to give specific feedback, students "really just want to say, 'All right. Did she give me a B? What am I getting here?'"

To more closely connect student motivation to assessment processes, teachers (n=53) described the ways in which they helped students self-assess and resubmit their programming projects until they were satisfied with their work, as well as the teacher's assessment of their work. Evan spoke about how teaching kids to self-assess helped them "monitor their own progress," and Betsy felt that self-assessment involved students as "participants

in the process." Typically, teachers were asking students to give themselves a grade and offer a justification of why they deserved that grade. A high school teacher, Jesse also scaffolded students towards self-assessing their final project by using a consistent rubric across multiple projects before giving students a blank template and asking them to define their final project's minimum requirements, saying, "What I'm looking for from them is to not only decide what they want to make, but also to decide how they want me to assess it." This process of self-assessment helped teachers gain a better sense of whether students felt that they had met their own goals, and how students' understanding of their own learning was connected to their teachers' assessments. Students were fairly accurate, or even too harsh, in their self-assessments. Natalie, a high school teacher, said, "Usually, when students grade themselves, they're either right at where I graded them, or they're within one grade. When I give them back my rubric with the feedback on it, they can see … what they would need to do to change it if they wanted to."

Some teachers (n=21) encouraged regular resubmission of work for full credit if students were dissatisfied with their grade and wanted to resubmit. However, resubmission sometimes worked against students—Justin, a high school teacher, pointed out that students could introduce new bugs, saying, "Sometimes, when they fix one thing, they break something else." Resubmission offered opportunities for students to demonstrate what they knew, rather than what they were able to do in any particular moment. Rhonda noted, "Anytime they don't do as well as they can, I give them a chance to retake … It's not a test of, 'Did they do it right the first time?'" In creating opportunities for students to be more involved with the assessment process, teachers were helping students deepen their personal investment in their programming projects.

Surfacing process through documentation and conversation

When teachers have greater understandings of students' project creation process, they are better able to support students' current and future learning. Across grade levels, teachers experienced difficulties in understanding individual students' learning solely by examining students' work product. Several teachers (n=12) noted that students could copy and paste code into their projects without understanding how the code worked. Others (n=35) raised that seeing a student use a concept in a particular project did not indicate whether they understood the concept and could transfer that knowledge elsewhere. Many teachers (n=38) designed programming projects to be collaborative, but it was difficult to understand individual contributions. Natalie, a high school teacher, described how students with greater expertise could easily dominate in group projects. When looking at students' code, teachers (n=49) had difficulties seeing what students cared about and how much effort they invested, what their "individual gains and personal bests" were (Richards, 2010, p. 193). Even with recent increased access to computing in K–12, teachers (n=36) described how some students had many years of prior computing experience, while other students were new to the subject. Brooke, a middle school teacher, described this differentiation challenge as one of equality, of "how to assess [students] equally based on their skill level, because they're all at such varying levels when they come in." In contrast, Janet, a K–12 technology integration specialist, aimed to differentiate her assessments, saying, "If I'm looking at the work of one student, and then a different student's work, I have to remember where this person started from." Without understanding learners' process, it could be challenging for teachers to celebrate students' successes and highlight areas of future learning.

Sawyer (2021) notes that "ideas do not originate from internal thoughts during solitary meditation; rather, they emerge from a visible and embodied process of engaging with the work" (p. 17). Most teachers (n=74) found ways to attend to students' process as well as product, asking students to articulate their ideas through written documentation or in-class conversations. A third of teachers (n=29) asked students to document their process as a way of making their thinking and learning visible. Documentation helped teachers gain insight into whether and how much students understand the code in their projects. Students wrote in-line comments in their code, pasted screenshots of their projects into separate reflection documents, or recorded summative video reflections. Adrienne, a middle school teacher, said, "You could copy code quite easily … which has been an issue … that's why I use the design process of creating documentation. You're able to talk about what you've been doing. You can explain how you got the information you got."

Documentation practices also helped students develop insight into what they had learned. In Evan's inclusive classroom, students did not always finish their projects. He noted, "I hate for students, especially [those] who have struggled or who have IEPs[1] that really presented challenges in their learning to think the project was a wash." Writing a final reflection helped students realize, "Hey, there are times where I hit a challenge … and I grew." However, several teachers noted that documenting process could take time away from the product, particularly when there was emphasis on written reflections for English language learners or students with disabilities. Shannon, a high school inclusion teacher, said that documentation could be "way, way too oppressive, and then they can't be creative because they're so busy writing [things] down. It's a fine line." Lindsay, a middle school teacher, said that working "with the population of learners who struggle with reading and writing, I can't say I'm surprised that they're not putting comments in between their lines of code to express their thinking."

Many teachers (n=68) described a desire to assess process by talking to students about their work, which helped to develop shared understandings of what was new or exciting. But this could be time-consuming. Damien, a high school teacher, wanted to have conversations with every student but said, "I don't ever do that.… I don't know how to make it happen." Robin, an elementary school teacher who saw every class every week, said, "I'd love to [meet with students], but I have 800 students." Stacy, a middle school teacher, shared a story about the challenges of not being able to see what students had learned through working on their projects.

> Several years ago, I had this student. This girl, who was just really fun and a very creative girl, and she turned in this game that was a cute game. I wrote, "This is a cute game. You did this well, and you did that well and it … meets expectations." I got this seething email from her father that has really stayed with me. He said, "You know what? My daughter wanted to be a computer programmer until this moment, and now she never wants to do it." I thought, *Oh, my God. What have I done?* "She spent so long on this one feature, and you did not notice this feature."

Because of this experience, Stacy changed her assessment practices to focus on deepening her understanding of her students and their process, telling them,

> Please tell me what you're really proud of in this. Please tell me what I might not notice. I want to know what you were up against, and what battle you won. And if you didn't win the battle, but you were fighting it, I want to know that too. Please just tell me what to pay attention to.

When teachers were able to learn more about their students' process, they were better able to support students in making progress on their programming projects. Teachers (n=68) relied on their direct experiences with students to determine what they had learned. Jill, a high school teacher, emphasized the importance of classroom observation: "I observe my students interacting with each other's projects acutely, because that says so much that I could never [understand] otherwise."

Discussion and conclusion

Creating programming projects can offer opportunities for students to engage in personally authentic computing experiences, which can support sustained participation in computing. Across contexts, we saw K–12 computing teachers aspire to support students' programming projects by encouraging possibilities, inviting multiple perspectives, increasing students' personal investment, and surfacing students' process. The process of learning to program by creating projects, however, is seldom predictable or linear, and students' diversity of interests often led to a diversity of learning outcomes. We saw teachers grapple with these complexities as they used various assessment practices. Their assessment practices were typically formative, helping teachers assess what students had learned thus far while supporting their future learning. Through *rubrics*, teachers were describing to students what they might try in their projects. Giving and receiving *feedback* helped students develop judgment of their own and others' work. *Self-assessments* helped students set their own goals and decide what they wanted to learn. And by *documenting and talking about their work*, students stepped back and articulated their thinking, which could advance their learning. Teachers rarely relied on a single assessment practice—different practices were used at different moments during students' project processes. When used together, assessment practices supported teachers in developing better understandings of their students as thinkers, creators, and learners.

However, teachers' desired assessment practices could be hard to implement, given the time constraints of K–12 schools. Teachers offered opportunities for students to resubmit work, but that could take time away from the next project. Some teachers said that there was insufficient time for peer assessment, and others pointed out that students needed time to learn how to give feedback. Notably, many of the elementary school teachers, who often saw more students for shorter class periods, cited time as a limiting factor. Teachers' assessment practices were also striving to address this particular moment in U.S. computing education. Because of computing education's uneven implementation across K–12, the ability to offer learning experiences that meet the needs of all students can feel daunting, and almost half of participating teachers (n=36) described challenges they faced in offering differentiated teaching and assessment. Teachers are often working in inclusive classrooms, where students may have a range of learning needs, and students are entering classrooms with a range of prior programming experience. Some students may not have access to technology at home, reducing the amount of time that students can work on their projects. Building relationships helped teachers gain a better sense of students' prior experiences with computing and how much effort students were investing in their projects, but this could be a time-intensive process.

In addition, a desire to understand student learning can have unintended consequences on the design of opportunities to create personally authentic programming projects. Because it can be challenging to surface what students have learned through examining their projects, there may be an impulse to assess what is easily visible, focusing on accuracy in usage of programming concepts and/or easily-checked solutions to programming puzzles. Sometimes, this can mean divorcing the assessment of learning from the project process, or this can mean constraining student agency in their projects, such that students undertake similar project creation activities. When we limit ourselves to what is easily assessed, we risk artificially constraining learning opportunities in ways that benefit teachers and researchers, and not the students themselves. And when we limit student agency in their projects, we risk limiting opportunities for students to engage in expansive learning when learning to program (Wilson, 2021). As computing education becomes more widespread, there is increased need to consider not only whether students have access to computing education, but also what *kinds* of computing education opportunities are accessible. What are students being asked to do as they learn to program? And how might assessment practices support, rather than constrain, student learning?

Assessment practices can unintentionally undermine student confidence and their progress. But as teachers demonstrated, assessment practices can also encourage possibility, invite additional perspectives, increase personal investment, and surface creative process. Teachers' practices pushed back against traditional forms of assessment and encouraged understandings of assessment as an integral component of the project creation process. We hope that these teachers' assessment practices inspire researchers and teachers to expand access to opportunities for students to pursue personally authentic programming projects, which can enable students to express their ideas with code and see themselves as computational creators.

Endnotes

(1) For young people to receive special education services in U.S. public schools, they are given individualized education programs, or IEPs.

References

Barnes, N., Fives, H., & Dacey, C. M. (2015). Teachers' beliefs about assessment. In H. Fives & M. Gill (Eds.), *International handbook of research on teachers' beliefs* (pp. 284–300). Routledge.

Barron, B., & Darling-Hammond, L. (2010). Prospects and challenges for inquiry-based approaches to learning. In H. Dumont, D. Istance, & F. Benavides (Eds.) *The nature of learning: Using research to inspire practice* (pp. 199–225). OECD Center for Educational Research and Innovation.

Beach, R., & Friedrich, T. (2006). Response to writing. In C. A. MacArthur, S. Graham, & J. Fitzgerald (Eds.) *Handbook of writing research*. (pp. 222–234). The Guilford Press.

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field*. Google LLC.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, *23*(4), 561–599.

Bower, M., Wood, L. N., Lai, J. W., Howe, C., Lister, R., Mason, R., Highfield, K., & Veal, J. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, *42*(3), 53–72.

Braun, V., & Clarke, V. (2012). Thematic analysis. In H. Cooper (Ed.), *APA handbook of research methods in psychology* (Vol. 2, pp. 57-71). American Psychological Association.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *Paper presented at the annual meeting of the American Educational Research Association, Vancouver, Canada.*

Brennan, K. (2015). Beyond right or wrong: Challenges of including creative design activities in the classroom. *Journal of Technology and Teacher Education*, *23*(3), 279-299.

Brinkerhoff, J., & Glazewski, K. (2004). Support of expert and novice teachers within a technology enhanced problem-based learning unit: A case study. *International Journal of Learning Technology*, *1*(2), 219–230.

Brookhart, S. M. (2008). *How to give effective feedback to your students*. Association for Supervision and Curriculum Development.

Capraro, R. M., & Slough, S. W. (2013). Why PBL? Why STEM? Why now? An introduction to STEM project-based learning: An integrated science, technology, engineering, and mathematics (STEM) approach. In R. M. Capraro, M. M. Capraro, & J. R. Morgan (Eds.), *STEM project-based learning* (2nd ed., pp. 1–5). Sense Publishers.

Cornish, F., Gillespie, A., & Zittoun, T. (2013). Collaborative analysis of qualitative data. In U. Flick (Ed.), *The Sage handbook of qualitative data analysis* (pp. 79–93). Sage.

Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: A review. *ACM Journal on Educational Resources in Computing (JERIC)*, *5*(3).

Earl, L. M. (2012). Assessment as learning: Using classroom assessment to maximize student learning. Corwin Press.

Grover, S. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 269–288). Springer.

Haduong, P., & Brennan, K. (2020). Talking in pairs: Learning from and with teachers through artifact-based dyadic interviews. In M. Gresalfi & I. S. Horn (Eds.), *Proceedings of The International Conference of the Learning Sciences (ICLS) 2020: Vol. 4* (pp. 2369–2370).

Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. *10th Koli International conference on computing education research* (pp. 86–93).

Jones, A., & Moreland, J. (2005). The importance of pedagogical content knowledge in assessment for learning practices: A case-study of a whole-school approach. *Curriculum Journal*, *16*(2), 193–206.

Kleinert, H. L., Kennedy, S., & Kearns, J. F. (1999). The impact of alternate assessments: A statewide teacher survey. *The Journal of Special Education*, *33*(2), 93–102.

Lew, M. M., & Nelson, R. F. (2016). New teachers' challenges: How culturally responsive teaching, classroom management, & assessment literacy are intertwined. *Multicultural Education*, *23*, 7–13.

Lumadi, M. W. (2013). Challenges besetting teachers in classroom assessment: An exploratory perspective. *Journal of Social Sciences*, *34*(3), 211–221.

Maxwell, J. A. (2010). Validity: How might you be wrong. In W. Luttrell (Ed.), *Qualitative educational research: Readings in reflexive methodology and transformative practice* (pp. 279-287). Routledge.

National Academies of Sciences, Engineering, and Medicine (NASEM). (2021). *Cultivating interest and competencies in computing: Authentic experiences and design factors*. The National Academies Press.

Pedersen, S., & Hobye, M. (2020). Implications of assessing student-driven projects: A case study of possible challenges and an argument for reflexivity. *Education Sciences*, *10*(1), 19.

Richards, R. (2010). Everyday creativity. In J. C. Kaufman (Ed.), *The Cambridge handbook of creativity* (pp. 189–215). Cambridge University Press.

Sawyer, R. K. (2021). The dialogue of creativity: Teaching the creative process by animating student work as a collaborating creative agent. *Cognition and Instruction*, 1–28.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158.

Smith, J., Tessler, J., Kramer, E., & Lin, C. (2012). Using peer review to teach software testing. *Ninth annual conference on international computing education research* (pp. 93–98).

Wilson, N. C. (2021). Examining the impact of systemic tensions on agency and identity: The multiple positions of Reggie in production-centered, technology-mediated activity. *Cognition and Instruction*, *39*(2), 181–209.

Yadav, A., & Berges, M. (2019). Computer science pedagogical content knowledge: Characterizing teacher performance. *ACM transactions on computing education (TOCE)*, *19*(3), 1–24.

Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., & Clayborn, L. (2015). *Sowing the seeds: A landscape study on assessment in secondary computer science education.* CSTA.

Acknowledgments

We would like to thank the computing teachers who generously shared their experiences and assessment artifacts. We would also like to thank Emily Veno, Sarah Blum-Smith, and members of the Creative Computing Lab for support in analysis and feedback of early drafts of this manuscript. This work was supported through Google CS-ER award #93661905.