Instructional Science
https://doi.org/10.1007/s11251-023-09629-5

ORIGINAL RESEARCH

Reflections on sustained debugging support: conjecture mapping as a point of departure for instructor feedback on design

Zachary D. Ryan[1] · **David DeLiema**[2]

Received: 28 October 2020 / Accepted: 15 March 2023

Abstract
This paper articulates an approach to incorporating instructor feedback in design-based research. Throughout the process of designing and implementing curriculum to support middle school students' debugging practices in a summer computer science workshop, our research and practice team utilized instructor-generated conjecture maps as boundary objects, providing insight into the instructors' reflections on their classroom teaching. We develop an analytic tool for categorizing instructors' reflections on their conjecture maps, attending specifically to how instructors push back on design choices, whether by envisioning new mediating processes, introducing new connections, discussing new design features, articulating confusion/uncertainty, and/or presenting hopes and predictions. The tool is then applied to seven instructors' daily reflections over the course of four weeks of instruction, focused on three conjecture maps. Overall, the paper documents a range of tensions that instructors encounter when aiming to provide sustained debugging support to students and introduces a tool for understanding the detailed ways that instructors critique design conjectures.

Keywords Design-based research · Conjecture mapping · Collaboration · Debugging · Computer science education

Educational researchers pursuing research-practice partnerships (Penuel et al., 2015), including within design-based research traditions (Brown, 1992; Cobb et al., 2003; Collins, 1992; The Design-Based Research Collective, 2003), have recently emphasized the value and process of pursuing equitable collaborations between researchers and teachers (Bang & Vossoughi, 2016; Farrell et al., 2021; Gutiérrez & Vossoughi, 2010; Philip et al., 2018; Vakil et al., 2016). In these partnerships, when it comes to bridging learning theories and

✉ Zachary D. Ryan
zryan@iu.edu

David DeLiema
ddeliema@umn.edu

[1] Indiana University, Bloomington, USA

[2] University of Minnesota, Minneapolis, USA

learning designs through explicitly "hypothesized learning processes" (Cobb et al., 2003, p. 10), or design conjectures (Bakker, 2018; Sandoval, 2014; van den Akker, 1999; Akker, 2013), researchers and teachers sometimes collaborate to formulate predictions about the learning design and retrospectively reflect on how the design unfolds (David et al., 2019; Lee et al., 2022; Leonard et al., 2017; Penuel et al., 2016; Tobar-Munoz et al., 2017). In a continuation of this DBR commitment, our first contribution of this paper is focused on presenting an analytical tool that provides a lens for categorizing instructors' reflections on how their in-classroom experiences (mis)align with their design conjectures. We share details about the development of this analytical tool to make it tractable for researchers—working in similar or distinct disciplinary contexts from our own—to extend, modify, and apply the tool in studies focused on the detailed documentation of teachers' perspectives on design conjectures.

The second contribution of this paper is to demonstrate the utility of this analytical tool in the context of a design-based research study. The context of this study is an informal summer workshop with computer science (CS) educators and 5th–10th grade students (Dahn & DeLiema, 2020; DeLiema et al., 2020, 2022). In this workshop, the learning designers and researchers focused their collaborative work on supporting students' approaches to debugging broken code. Educational researchers have extensively studied debugging, including attending to common bugs (DesPortes & DiSalvo, 2019; Endres, 1975; Ripley & Druseikis, 1978), tools for debugging (Hristova et al., 2003; Ko & Myers, 2009; Lazar et al. 2017; V.C. Lee et al. 2018; Miljanovic & Bradbury, 2017), and strategies for debugging (Ko et al., 2019; Prather et al., 2019). However, there is by comparison much less work on how educators experience teaching debugging in the classroom. An emerging body of work in naturalistic learning settings attends to how teachers support students with debugging during classroom conversation (e.g., DeLiema et al., 2022; Dickes et al., 2020; Flood et al., 2018; Heikkilä & Mannila, 2018; Hennessey Elliott, 2022; Silvis et al., 2022), and some work examines teachers' reflections on how they implement and adapt instruction in the CS classroom (e.g., Shaw et al., 2020), but even less work covers teachers' reflections on debugging (Fields et al., 2018; Michaeli & Romeike, 2019). To grow this latter thread, our paper focuses on how middle school CS teachers describe their debugging instruction in daily reflections over a month-long summer workshop.

In all, our paper extends the CS education literature on teachers' experiences with debugging in the classroom and advances an analytical tool consistent with the commitment in DBR to value teachers as core contributors to the production and examination of design conjectures. Toward this end, we center the following research question: *Using instructors' reflections on their day-to-day experiences teaching programming, what can we understand about the challenges they face with respect to providing sustained debugging support?* In our analysis, we highlight eight themes that illuminate the day-to-day friction and improvised adaptations instructors report experiencing when providing sustained debugging support to their students. Amid the rapid growth in K-12 CS education (Webb et al., 2017), requiring educators to develop CS education expertise (Haduong & Brennan, 2019; Menekse, 2015), leaders to develop high quality professional development (Warner et al., 2019), and stakeholders to center inclusivity in CS (Sullivan & Gresalfi, 2020), the study we present here illuminates central problems of practice and improvised adaptations around debugging pedagogy that can ultimately inform both curricular and professional development in K-12 CS contexts.

Collaboration and teacher reflection in design-based research

In preparing, conducting, and retrospectively analyzing learning designs responsive to particular shortcomings, goals, and/or problems of practice, design-based research (DBR) departs from and aims to shape theories of teaching and learning (Cobb et al., 2003). Designated as design experiments for education (Brown, 1992), DBR is modeled after iterative design cycles in engineering research (Collins, 1992). By situating the research process in naturalistic teaching practices, DBR provides "a lens for understanding how theoretical claims about teaching and learning can be transformed into effective learning in educational settings" (The Design-Based Research Collective, 2003, p. 8). Over the past 30 years, DBR has increasingly served as a tool for learning scientists to engage in systematic, iterative research with the potential to directly affect change in learning environments as well as generate theoretical contributions (Bakker, 2018; Brown, 1992; Collins, 1992; Collins et al., 2004; Design-Based Research Collective, 2003; diSessa & Cobb, 2004; Edelson, 2002; Sandoval, 2004).

In recent years, educational researchers have called for greater attention to how power operates within DBR partnerships. Building on prior efforts toward this end, including social design experiments (Gutiérrez, 2008; Gutiérrez & Vossoughi, 2010; Gutierrez & Jurow, 2016) and formative interventions (Engeström, 2011; Bang & Vossoughi, 2016) argue that "design decisions in much of design research are typically made by 'experts' who inhabit privileged positions in the world and less often elaborate on efforts to engage in collaborative processes with practitioners, families, youth, or community members" (p. 174). Bang and Vossoughi (2016) articulate an alternative, expansive view of *participatory* design research that includes attending to power in multiple facets of the research process: processes of partnering, decisions to focus on particular interventions, resistance to dominant (but neutrally positioned) forms of cultural capital, inclusion of epistemic heterogeneity, attention to students' relationships with one another, and the quality of discussions about design. Educational researchers have delivered on this expansive vision of participatory design-based research in a number of ways, including by attending to trust in partnerships, the process of choosing the objects of design activity, and how labor is made visible and understood in collaborative design (e.g., Jurow et al., 2016; Vakil et al., 2016; Zavala, 2016). These issues of power in learning designs are particularly central to and yet underexplored in the space of K-12 CS education (Santo et al., 2020).

A recurring site of power in DBR are moments in which teachers and researchers reflect on learning designs as a means to critique and revise those designs (Cimer et al., 2013; Schon, 1987). We define reflection here as activities in which one engages in thinking about prior experiences in order to work toward new understandings (Boud et al., 1985; Dewey, 1933). Schon's (1992) constructivist approach sees the process of design as a "reflective conversation" with the materials of a design situation throughout each step in a design cycle. We take up a similar approach by utilizing what Schon (1987) calls *reflection-on-action*, or deliberately thinking about a specific action or event outside of when it occurs, as a way to sustain collaborative reflection over the duration of a project. This can be seen as a "systematic process of deliberation enabling analysis, reconstruction and reframing in order to plan for further teaching and learning" (Day, 1999; p. 28). Within the learning sciences, this kind of reflective design work has taken the form of collaboratively forming professional vision alongside instructors (Gomoll et al., 2022). Professional vision (Goodwin, 1994) refers to shared discursive practices used to understand and interpret an event informed by and situated within communities of practice. One example of this in

collaborative DBR is "video clubs" where teachers look over video footage of classroom interactions to unpack and reflect on what they observed alongside researchers (van Es & Sherin, 2010). Reflections-on-action of this kind supports the DBR cycle by inviting participants to think deeply about how the design was enacted in situ, which then informs future iterations of the design. In the spirit of these reflection practices, we sought in our study to have a form of sustained reflection-on-action in which instructors, on each day of their teaching, wrote about their experiences teaching debugging in their classrooms. As we detail in the next section, we structured this reflection process around collaboratively created graphic representations known as conjecture maps (Sandoval, 2014).

Conjecture mapping and recent emphases on collaborative dynamics

A central challenge in DBR is to generate theoretical insights from studies conducted in naturalistic learning contexts (Kelly, 2004; Levin & O'Donnell, 1999; Phillips & Dolle, 2006; Shavelson et al., 2003). DBR researchers often articulate (and iterate on) testable design conjectures that track connections between theories of teaching/learning and actual design decisions (Cobb et al., 2003). These approaches coordinate "learning processes and the means of 'engineering' them" (diSessa & Cobb, 2004, p. 83). Bakker (2018) highlights an early framework for composing design conjectures from van den Akker (1999, 2013) that includes the purpose/function of the design, the design's essential characteristics and procedures, and the rationale for the design, with respect to both theoretical and empirical dimensions. As Bakker (2018) notes, the format of these design conjectures "combines the *how* and the *why*, and thus allows the research to connect a value (*why* in terms of purpose) with actions (*how* in terms of design or procedures) underpinned by arguments (*why* in terms of scientific knowledge and practical experience)" (p. 49). This argumentative grammar ensures that conjectures are more than just predictions; they also entail formulating explicit connections between the design's backing and the design's features.

When articulating design conjectures, many educational researchers have gravitated toward using Sandoval's (2014) conjecture mapping framework (e.g., Bismack et al., 2015; Brown & Crippen, 2016; Danish 2014; Land & Zimmerman, 2015), including its visual template (Kali et al., 2015; Saleh et al., 2019; Tucker-Raymond et al., 2019; Wozniak, 2015). A prototypical conjecture map is made up of several elements. First, it includes a *high-level conjecture*, informed by theories of knowing and learning, regarding how to support students' learning toward valued ends. Second, the map describes how that high-level conjecture is *embodied* in the features of the designed learning environment (e.g., tools, activities, participant structures, and discursive practices). Third, researchers articulate short-term *mediating processes* expected to follow from particular features of the design. Finally, the map documents the research team's predictions about how these mediating processes will in turn produce *longer-term*, desired outcomes of the research. Conjecture maps delineate design conjectures from theoretical conjectures (Sandoval, 2004, 2014). *Design conjectures* refer to the part of the conjecture map that links the embodied design with mediating processes. In contrast, *theoretical conjectures* refer to the part of the conjecture map that details how the short-term, mediating outcomes link up with desired outcomes of the intervention.

We chose conjecture mapping to organize teachers' reflections because it is commonly used in the field of DBR, provides a transparent and accessible approach to the process, and has been used by researchers collaboratively designing with teachers. In practice, the

most common use of conjecture mapping is to have researchers handle the task of building and assessing conjecture maps (e.g., Bismack et al., 2015; Kali et al., 2015; Wozniak, 2015). For example, some researchers design conjecture maps for a project, and use incredibly rich data and analyses to inform the efficacy of those conjectures (including iterations across stages of the design), but do not involve the participants in the examination of the conjecture maps themselves (Bland et al., 2017; David et al., 2019; V.R. Lee et al., 2018; Stromholt & Bell, 2018; Tobar-Munoz et al., 2017; Wilkerson, 2017). However, as educational researchers gravitate toward collaborating with teachers in the process of design (Matuk et al., 2016; Penuel et al., 2007; Sanders & Jan Stappers, 2008; Severance et al., 2016), including attending to how power influences design in teacher-researcher partnerships (Gutiérrez et al., 2016; Nasir & Vakil, 2017), some design-based researchers have started to invite teachers into the process of working with the conjecture maps themselves. In one case, the research participants leading the design efforts were invited to examine the conjecture maps the research team made as "a visual means…to engage in this reflective examination" (Bland et al., 2017, p. 5). In yet other efforts, teachers were directly involved in the construction of the conjecture maps either as collaborators in design workshop activities (Lee et al., 2022; Penuel et al., 2016), or as members of the research team involved in all parts of the design research (Leonard et al., 2017). We have aimed here to combine these latter approaches by asking instructors to reflect in a sustained way on conjecture maps that they developed, and we extend this work in part by building an analytical tool to document instructors' observations about design conjectures.

Design failures as focal points

DBR brings together participants with disparate points of view, and through sustained, collaborative engagement with design and evaluation, participants work toward embracing and reflecting deeply on the divergences and uncertainties that occur at the boundary between each participant's perspective. In this process of boundary *crossing* (Penuel et al., 2015), the artifact of a conjecture map serves as a boundary *object* (Star & Griesemer, 1989; Suchman, 1994). The conjecture map makes a collective conjecture transparent to the research and practice community and allows each member to voice comments and concerns on its traction during teaching and learning. Because design-based researchers attend to what Cobb et al. (2003) describe as placing "theories in harm's way" in order to "capitaliz[e] on contingencies that arise as the design unfolds" (p. 10), the conjecture map serves as way to gauge how the unfolding of the design clashes with a team's initial predictions and assumptions. Given that moments of breakdown or failure can serve as points of departure for learning (Engeström, 2001; Kapur, 2008), it is particularly important to embrace teachers' and researchers' observations about discontinuities between the unfolding design and the team's design conjectures. Teachers and researchers' ongoing observations about the efficacy of a theoretical or design conjecture could signal a point at which the design is falling short, in perhaps unexpected ways (Penuel et al., 2015; Suchman, 1994). For these reasons, in our documentation and examination of instructors' reflections on their conjecture maps, we pay particular attention to occasions in which instructors discussed adapting designs, seeing students' engagement with the design that clashed with initial design conjectures, and/or observing heterogeneous student engagement with the design.

Instructors' experiences teaching debugging

Our effort to demonstrate the utility of this analytical tool is centered in the context of CS education and specifically focuses on the process of debugging. The debugging process—comprised of students and teachers iteratively noticing a *problematic deviation* from preference or expectation, proposing a *cause* of the deviation (often called a "bug"), and attempting to *intervene* (DeLiema et al., 2021; Ko & Myers, 2005; Spohrer et al., 1985)—provides a promising focal point for this research. One benefit is that because debugging is an inescapable part of the practice of coding and the process of learning to code, there is a considerable body of scholarship on the topic (e.g., McCauley et al., 2010; Zeller, 2009). The majority of this prior work attends to students' experiences learning to debug, such as common bugs that students introduce into their code (Endres, 1975; Ripley & Druseikis, 1978), tools for debugging that are built into programming environments (Hristova et al., 2003; Ko & Myers, 2009; Lazar et al. 2017; V.C. Lee et al. 2018; Miljanovic & Bradbury, 2017), and strategies or processes for debugging (Ko et al., 2019; Prather et al., 2019). There is also evidence that instructors play a role in students learning how to debug (Klahr & Carver, 1988; Michaeli & Romeike, 2019). There are even arguments for "debugging first" approaches to CS instruction (Lowe, 2019), including explicit guidelines for how to approach debugging in K-8 curriculum (Rich et al., 2019).

However, the existing research literature is considerably less focused on what instructors experience in their day-to-day engagement with teaching CS. The studies that have addressed this topic have recognized that programming content knowledge is an important but insufficient resource (Liberman et al., 2012), and that beyond content knowledge, new instructors struggle to handle the student-centered demands of the practice, including keeping up with one-on-one instruction, supporting students' unique problem solving approaches, and assessing skill in collaborative participation structures (Yadav et al., 2016). In addition, Shaw et al. (2020) document teachers' recruitment of local resources in their efforts to center inclusion in an open-ended CS unit, and Ryoo (2019) engaged in a series of teacher interviews that asked instructors to reflect on the practices they identified as particularly effective in their CS classrooms, which included ensuring that students' voices and perspectives were valued in the classroom community. In the specific context of debugging, some scholarship has documented social interaction between teachers and students during debugging (e.g., DeLiema et al., 2022; Flood et al., 2018; Hassenfeld & Bers, 2020; Heikkilä & Mannila, 2018; Hennessey Elliott et al., 2022; Silvis et al., 2022), but this work attends to the public record captured on camera and audio recordings, not teachers' personal reflections. In addition, researchers have examined how new CS instructors debug code on their own (Kim et al., 2018; Vasconcelos et al., 2020), studies that are key to informing teacher professional development but do not directly examine day-to-day debugging support. On the other hand, at least one workshop focused on how teachers experience the process of teaching debugging was held at an annual CS education research conference (Lewis & Gregg, 2016). In addition, Michaeli and Romeike (2019) recently articulated the need for this area of research: "To eventually develop suitable approaches, best practices, and materials for the classroom, we have to incorporate teachers' existing experience as well as their personal perspectives towards debugging. As teachers are confronted with students' programming errors *on an everyday basis*, we want to investigate their approaches and best practices" (p. 1030, our emphasis). Their work documented how instructors "cope" with debugging instruction, including how instructors perceived students' reactions to bugs, students' coding skill, and the quality of students' requests for debugging support

(Michaeli and Romeike, 2019). Toward this end, albeit with less emphasis on debugging, Fields et al. (2018) focused on two instructors' "emergent practices" in an electronic textiles unit and noticed how instructors position students as having expertise and draw connections across student designs.

We extend these prior efforts in the current study in a few ways. First, we extend the field's recent commitment to centering teachers' *experience* of the CS classroom by focusing on how they reflect on their efforts to support debugging. Without this work, the design of debugging tools, strategies, and research cannot be informed by the very people who most provide debugging support in the classroom. Second, we examine teachers' reflections in a longitudinal design that involves two weeks of instructional planning and a month of CS instruction. By gauging multiple instructors' reflections on a daily basis, we are able to capture a wide range of their experiences as students' programming skills and coding projects developed. Third, by situating teachers' reflections in a DBR study in which the teachers developed design conjectures and retrospectively analyzed these conjectures, our approach positioned teachers' observations and reflections as a central data source and offered a systematic conjecture mapping structure to organize their observations.

Method

Context and participants

This research took place at a non-profit community learning center in a large city on the west coast of the United States. The non-profit provided free weekend workshops and two-week summer workshops to students interested in computer programming and who had signed up in advance. Part of a larger DBR project (DeLiema et al., 2020), this data collection effort took place within the first year of the DBR partnership and at a stage where our first pedagogical approach (described below) was beginning to emerge but where we had explicitly planned to closely examine and iterate on this pedagogy in years 2 and 3 of the project. Extending our prior work in this DBR partnership that documented growth in students' debugging confidence and awareness of debugging strategies (DeLiema et al., 2020), case studies of students' generative reflections on problem solving, emotion, and identity (Dahn & DeLiema, 2020), and case studies of students' pursuit of multiple valued debugging processes (DeLiema et al., 2022), here we set out to center *instructors' experience of teaching debugging*, namely the moments of friction, difficulties, and ongoing adjustments they report in their teaching that can inform future designs of debugging tools, strategies, and pedagogies.

Seven paid instructors participated in the research. Three identified as male, and four identified as female. Six were working toward undergraduate CS degrees. The other instructor was a working professional who had the summer free to teach in the program. Instructors worked in pairs in classrooms with about 20 students. All of the instructors had experience as programmers, some shared their prior experience as small-group (e.g., one-on-one) CS tutors, and all had participated in a CS education professional development workshop at the non-profit. Many had also taught in the non-profit's weekend workshops for students ahead of this summer study. However, all were in the early stages of learning to teach in whole-class CS education spaces with 10–15 students.

Students ranged in grade level from 5th grade–10th grade, and either demonstrated need for financial aid or attended a school that serves a high percentage of low-income students.

Students were grouped based on age and programming expertise. One classroom was composed of 5th graders who had not previously learned any coding, two classes were composed of 6th–8th graders (some of whom had prior experience programming), and one class was composed of early high school students who had some prior experience programming. Most students in the workshop identified as Latino/Latina, and small proportions of students identified as Asian, Black, and White, including students who identified with multiple categories. Students identified as male and female in roughly equal proportions in each classroom. 63 students attended the first two-week workshop and 60 students attended the second two-week workshop.

Design of summer CS workshops

This paper focuses on data collected during back-to-back two-week summer CS workshops in 2017 (75-minute classes, 3 classes per day, Monday–Friday instruction, resulting in about 50 h of instruction), which we refer to for the remainder of the manuscript as *summer CS workshops.* The summer CS workshops included a range of project-based coding contexts: abstract cubist drawings of animals in PixelBots; LEGO mindstorm robot battles; LEGO mindstorm robot dances; a video game in OpenProcessing based on Agar.io, in which circle avatars consumer one another and grow larger; building construction in Minecraft; and beat machines in OpenProcessing. Each group of students participated in three of these project-based contexts during their two-week session. The curriculum involved students learning the techniques core to the project on small practice exercises before students applied these techniques in their own projects, working on them over several days. In a typical lesson, instructors introduced students to a new coding concept through whole-class modeling and discussion; students then practiced these techniques on a series of coding challenges independently and/or in small groups; and then over several days implemented these techniques in a custom final project (e.g., a robot dance or video game). The workshop focused in large part on students developing competence with debugging through the activities described above. In addition, students reflected on their experiences learning to code through arts-based activities (Dahn & DeLiema, 2020; Dahn et al., 2020). Breaks and a long lunch were interspersed between coding and art classes. On the final day of the workshop, students' families gathered at the learning center for a meal and a chance to explore students' projects.

Learning theories undergirding the design of the summer CS workshops

The learning theories that most undergirded the design of the curriculum merged principles of constructionism (Kafai, 2006) and sociocultural theory (Vygotsky, 1978). On the constructionist side, students coded projects they designed themselves, using programming environments that resembled Papert's (1980) original Logo platform. On the sociocultural side, students learned new programming skills and refined their projects in the context of consistent tool-mediated interactions with peers and experts. Reciprocal teaching (Palinscar & Brown, 1984) and known debugging strategies (e.g., McCauley et al., 2010) further informed our teaching practices and inclusion of step-by-step debugging instruction (see details in the next section). However, in focusing on supporting students' debugging growth, identities, and emotions as programmers through in-classroom scaffolding and arts-based reflections (Dahn & DeLiema, 2020; Dahn et al., 2020), this approach to learning design problematically eschewed *contrapuntal* considerations of inequities in the

professional programming workplace and considerations of the impact of modern technologies on society (Philip & Sengupta, 2021).

Collaboration on the development of conjecture maps

In the half year before the summer workshops, the non-profit hosted winter and spring weekend workshops (eight sessions each). In winter and spring, four of the instructors worked alongside the workshop's coordinators and the research team to conceive, pilot test, and refine pedagogical practices surrounding curriculum that had been written by the non-profit. These pedagogical practices were meant to inform how teachers introduced and closed out lessons, and how teachers collaborated with students when they were debugging their programs, both of which were underspecified in the provided curriculum.

In the 2 weeks just before the start of the summer workshops, all of the instructors participated in planning and professional development (PD). As instructors familiarized themselves with the curriculum, they would be using that summer, the instructors collaborated with a researcher (the second author of this paper) across three 2-hour sessions to solidify their pedagogical approach to debugging and generate conjectures about how their approach might work in their classrooms. In the first of these planning sessions, the team decided to focus on three pedagogical practices. The first pedagogical practice involved asking students to write daily, journal-based reflections on their debugging goals, strategies, and past debugging experiences, including using hashtags to capture emotional valence and sharing reflections in whole class dialogue, both at the beginning and end of each class session. The second involved instructors modeling debugging processes in one-on-one sessions with students, followed by prompting students to try those debugging strategies, and eventually listening as students narrate their own debugging process. The third approach described what instructors would model: a multi-step process of debugging, including getting in a debugging state of mind, comparing the intended code outcome with the actual code outcome, proposing causes of the bug, intervening to address the bug, and reflecting afterward (DeLiema et al., 2020).

After deciding to focus on these three pedagogical practices, the group then collaborated to create conjecture maps for each practice. Our approach resembled that of the Research + Practice Collaboratory (Penuel et al., 2016), where practitioners from the Council of State Science Supervisors and educational researchers collaboratively visualized design conjectures around how to support research and practice collaboration in the context of science teaching and learning. Similarly, our approach aimed to provide opportunities for instructors to develop conjectures about how students would respond to each of the three focal pedagogical practices. The conjecture maps acted as a collaborative tool for the researchers and instructors to "increase sensitivity to the various interactions possible with different elements of the designed learning environments" (Leonard et al., 2017, p. 13). The process of developing these maps involved a researcher broadcasting an empty conjecture map on a large screen in front of the group, seated around a table. Each instructor then took a turn describing details they wanted to add to the map. As each instructor offered descriptions of the pedagogical practices, hypothesized short-term outcomes, and valued long-term outcomes, the researcher typed their words into the screen-projected conjecture map. The group iterated on each conjecture map over the course of three days. The instructors produced the language in the map, revised the language, negotiated sticking points, and committed to a final version of the design conjectures. For this reason, we refer

to these artifacts as instructors' conjecture maps[1]. In the "Discussion" section of this paper, we consider how we as researchers might have better attended to (and mitigated) power dynamics among the DBR team at this key juncture of pedagogical planning.

Surrounding the process of developing design conjectures, the instructors spent time noticing facets of the debugging process in video data of students and instructors from the earlier winter and spring workshops (e.g., Sherin & van Es, 2005; van Es et al., 2017) and practiced their own debugging pedagogy in teacher-student role plays, a generative professional development technique (e.g., Lampert et al., 2013). These experiences grounded the conversations that took place around the design of the three conjecture maps. At this stage, our research team did not ask the teachers to visually document lines (or connections) between design features and hypothesized short-term outcomes. This decision, which simplified the traditional conjecture map template, made it possible for instructors to fill in observed connections as they worked with the pedagogical practices in their classrooms throughout summer.

Teacher-led retrospective analyses of the conjecture maps

Following the collaborative conjecture mapping process, the workshop's leaders and the research team invited the instructors to try out these pedagogical practices whenever the instructors deemed relevant throughout their instruction and revise them in whatever way they believed might improve them. Building on our prior work demonstrating that students at the end of the workshop perceived personal growth in their skills for handling bugs and gains in confidence during debugging (DeLiema et al., 2020), and consistent with our research question focused on the challenges instructors experienced when providing debugging support to students, we sought in the current study to better understand teachers' reflections on their debugging instruction.

During the 20 days of the summer CS workshops, instructors dedicated a half hour each day during their break from teaching to reflect on one of the three conjecture maps. We asked instructors for daily reflections in order to gather granular and longitudinal data on how instructors were thinking about their focal pedagogical practices. Engaging with daily reflections also ensured that instructors were reflecting on classroom activities they had recently experienced. Each instructor was randomly assigned to comment on a particular conjecture map on each day (e.g., on day 1, two instructors reflected on conjecture map 1, two instructors reflected on conjecture map 2, etc.). This was a practical decision because instructors only had so much time to retrospectively analyze and write about their conjecture maps during a busy day with students. By spreading instructors across maps, we guaranteed that at least one instructor would be commenting on each map each day, and that over time, each instructor would have multiple opportunities to comment on each map.

Instructors worked alone when reflecting on each map, giving them a chance to share their observations without the added complexity of peer input (e.g., Vedder-Weiss et al., 2018). During these reflection sessions, instructors looked at an electronic copy of the map. To guide their thinking about these maps, the research team designed a survey that prompted instructors to reflect on how they implemented the design in the classroom, what (if any) anticipated short-term outcomes arose, what they saw in the classroom to support those inferences, what other ways they saw students respond to the designs, and

[1] See the "Limitations" section for our reservations about this decision.

how the instructors wanted to adapt the designs moving forward (see Table 1 for complete survey questions). We viewed each of these as instructor-led "debriefing sessions" about design implementations (Cobb et al., 2003, p. 12), part of micro design cycles in which each "micro-design cycle consists of an anticipatory thought experiment, the enactment of instructional activities, and analysis, leading to adaptation or revision of subsequent activities" (Prediger et al., 2015, p. 879). Following the summer CS workshops, the research team interviewed teams of instructors about their experiences implementing and revising the focal pedagogical practices; their responses served as a point of triangulation for our data analysis approach (detailed later).

Developing a conjecture mapping analytic tool

By the end of the summer CS workshops, each instructor had provided multiple written reflections on each conjecture map, and it was evident to us that the depth of their reflections warranted a close look. The nature of the instructors' positions in the workshop meant that they were not readily available for continued in-person collaboration once the workshop concluded. With a commitment to preserving as much as possible the voices of the instructors in our analysis, we aimed to create an analytical tool to unpack instructors' reflections, and then apply the tool to aggregate key themes. This analytical tool offers a lens that researchers can bring to data of instructors' reflections on conjecture maps. More concretely, the analytical tool describes common ways that instructors reflected on conjecture maps. By attending to these common reflection practices, researchers can hold themselves accountable to a range of ways instructors think about design. For us, this analytical tool helped in our process of noticing commonalities in instructors' reflections on teaching debugging. Given the utility of the analytical tool, we planned in this paper to share details about its development and implementation to make it more tractable for research teams to extend, modify, and apply it in other DBR settings.

We started developing the analytical tool by aiming to understand the types of comments instructors formulated about their conjecture maps; this was less about *what* instructors communicated than about *how* they communicated about the conjecture maps. We refer to these as *reflection practices* throughout this paper. The multimodal setting for instructors' reflections—visual conjecture maps and instructors' typed statements—nudged us to consider the artifacts and documents used during the reflection activity (Derry et al., 2010; Jordan & Henderson, 1995), echoing commitments in situated cognition and multimodal interaction analysis (e.g., Goodwin, 2018; Hall & Stevens, 2015). Similar to artifact-based interviews, video-cued recall, and video-cued ethnographies (Adair & Kurban, 2019; Brennan & Resnick, 2012; Tobin et al., 2009), we viewed instructors' written reflections less as decontextualized remarks than as reactions to and clarifications of features of the conjecture maps. For these reasons, our development of the analytical tool blended the bottom-up approach of the *constant comparative* method (Glaser, 1965) with top-down constructs central to conjecture mapping, such as how instructors reflected on design features, mediating outcomes, and causal connections seen within their classroom practice (Sandoval, 2014). We aimed to foreground *multivocality*, or what Tracy (2010) describes in part as researchers' efforts to "not put words in members' mouths, but rather attend to viewpoints that diverge with those of the majority or with the author" (p. 844). For our qualitative work here, this meant persistently referring back to instructors' language when developing our analytical tool.

Table 1 Reflection prompt questions and a sample of instructor responses

Questions				
How did you bring this design to life in your classroom?	Did the design lead to any of our expected short-term outcomes? Why do you think this happened?	What are you seeing happening in your classroom to make that determination?	What were the unexpected ways students responded to this design choice, both positive and negative? Flesh this out to provide some context.	How might you adapt this design moving forward to make it better reach our valued short-term outcomes? Why might this work?
Instructor ST Day 3				
With the intermediate students, I don't model debugging every time (but I do with students who have less experience), I mostly try to prompt them to explain their bugs, then listen to their thought process.	I did see some students propose their own solutions to their bugs; at the very least most students were able to answer specific questions regarding the location of the bug, error messages, the API, etc. (responding to my prompts, essentially)	Students are able to answer my questions about their bugs more often than not.	Most students respond positively to being prompted (I think because they know it will lead to an eventual fix); there are a few occasions when it seems students feel frustrated after a series of prompting questions - this may be due to a lack of experience, at which point I would model it for them.	I want to start prompting more and modeling less in my classes; from my experience today, sometimes I try leading students in a certain direction, but they will propose an entirely different course of action which is something I definitely want to encourage.
Instructor SS Day 4				
Our server kept crashing so we tried to have them do it on the board. While writing the program, we purposely put in a bug and also had 'bugs' when the students wrote commands on the board. Then we went through the debugging process together as a class, where we went through the code line by line and had the students find and fix the bugs.	We saw alot more student voice when we let them write the commands on the board themselves. They also were able to find many of the bugs on the board themselves, without prompting from us.	More student voice in answering questions. Many students raised their hands when they spotted a bug.	No unexpected ways.	Have them narrate the debugging process in their own code when they find a bug. Asking them to reflect on their goals, the process we modeled for them and step through the code with us. It might work because it will get the students in the habit to automatically start doing it first.

Table 1 (continued)

Questions				
Instructor DT Day 5				
When encountering a debugging moment with a student, at first we pretty much model the process for them, changing things for them on their computer. We show them strategies for debugging and tools we use, and explain their purpose. In later instances, we prompt them by reminding them of their tools and strategies, and eventually we see students debugging on their own.	Yes, all of these short-term outcomes are visible in the classroom. This probably happened because of sheer repetition, as well as just increasing familiarity with our debugging practices. Instead of having to freak out when they see a bug, students have some kind of plan they can follow.	I see students using our debugging vocabulary and using it to describe their issues. They use tools without even being asked, and even debug their own issues independently. When they ask for help, they're able to pinpoint what's probably wrong and their questions usually lean towards unfamiliarity with the new content of the lesson.	Some students still find the debugging processes we give them foreign and must be reminded each time. Other students just want to code and get easily frustrated by bugs, and don't want to take the time to step back and use some of the strategies we give them. Today, we had an especially difficult challenge for students, which got them all frustrated to the point of wanting to give up. They did have some good reflections on it, which hopefully we can address tomorrow.	I think the conjecture map makes a lot of sense and doesn't need much revision. However, trying to help students who reject the debugging processes become more independent coders seems to be a point of struggle and I'm not sure how to fix that.

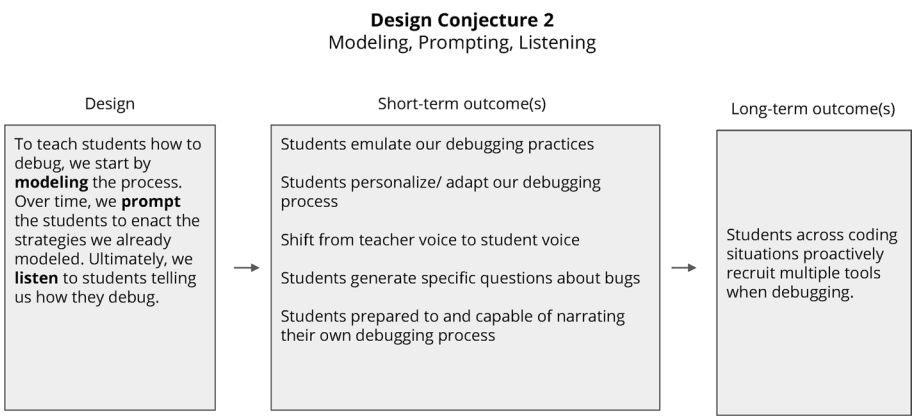

Fig. 1 One example of the conjecture maps used for the Debugging failure project

We enacted the above commitments first by examining two instructors' remarks about one conjecture map from one day of instruction. We enacted an open-coding approach (Khandkar, 2009) to this data analysis in order to find initial trends and patterns in instructors' reflections. Consistent with other open coding efforts (e.g., Lee & Dubovi, 2020; Nasir & Cooks, 2009; Vea, 2020), we used open coding in the early stages of our qualitative analysis to create a "descriptive, multidimensional preliminary framework for later analysis" (Khandkar, 2009, p. 8). In this initial stage of analysis, Author 1 looked at a single response written by an instructor for a single day's reflection, categorized *how* the instructor reflected on the specific conjecture map (keeping their quoted statement attached to our inference), compared their response to the original map, and then attempted to transpose the statement visually onto the conjecture map. Over a two-month stretch, Author 1 successively open-coded this instructor's reflections line-by-line and brought it to a weekly research meeting to discuss the emerging codes with Author 2. In these sessions, we noted moments of disagreement between team members and returned to the data in order to refine our description of reflection practices. Once disagreements were settled with this small sample, we then expanded our focus to four instructors (a subset of this data can be found in Table 1). Each research team member separately coded two instructors' reflections, before we reconvened and again settled any disagreements in the emergent open coding process.

To keep instructors' remarks within the context of the conjecture mapping artifact, we visually represented their written reflections on top of the original map (see Fig. 1). As candidate reflection practices emerged in our analysis, Author 1 developed and refined a set of representational conventions for them using color (now symbols in Fig. 2), shape, and line styles. We moved deliberately over several months through this sample to help ensure that instructors' voices were preserved and that we had not missed key reflection practices. As we overlaid more teacher reflections onto the conjecture maps, we noticed a number of common reflection practices. Instructors *referenced* existing features of the map without adding new detail (a triangle), they *unpacked* how a design or short-term outcome unfolded in the classroom with new details not in the original conjecture map (a diamond), and they *created* new features in the design and short-term outcomes sections (a circle). In addition, we abstracted three evidential markers instructors used when referencing, unpacking, and creating. Instructors described events they *observed/experienced* (straight lines),

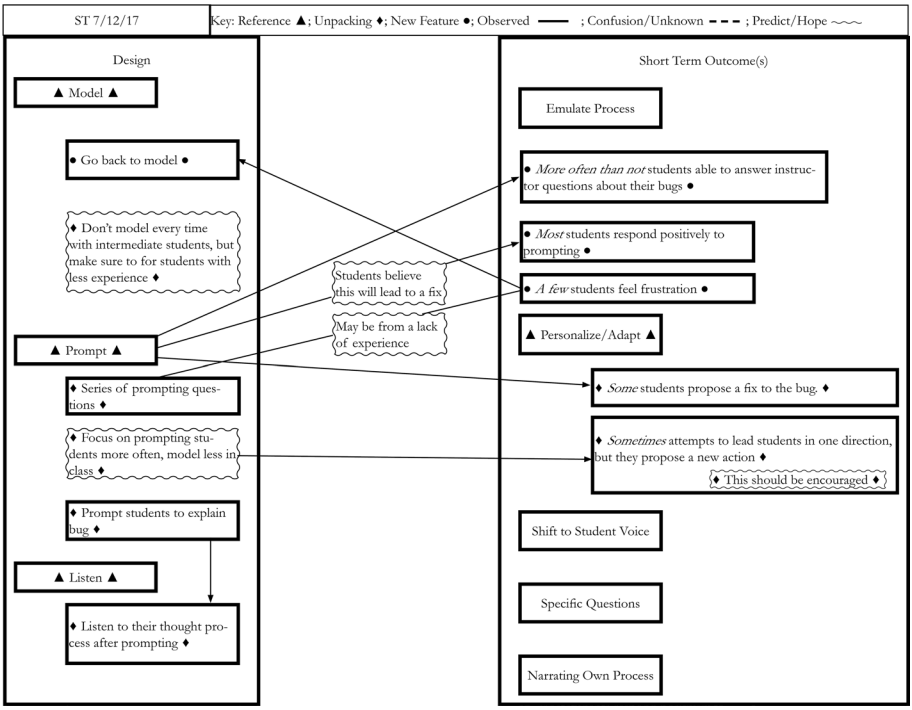

Fig. 2 Conjecture maps with Instructor feedback implemented

events they *wanted/desired/predicted* (wavy lines), and events they judged to be *confusing/unknown* (dotted lines). These modified conjecture maps (see Fig. 2) served as a representational tool for categorizing the precise moves and predictions instructors made surrounding the learning design.

Our analysis further recognized that instructors posited *intermediary outcomes* beyond what had been proposed in their original conjecture maps. These intermediary outcomes described mediating processes that instructors used to explain why the designs led to short-term outcomes in some cases but not in others. Instructors also proposed *backward connections* moving from the short-term outcomes back to the design when describing how they adapted to unforeseen circumstances. The product of this initial analysis process was an analytical tool designed to unpack and categorize how instructors reflected on their practice in relation to their conjecture maps. Our approach aligns with how open coding is carried out in qualitative analysis in which preliminary codes are refined into categories, which we marked as reflective practices, and then later applied to a larger data set and/or triangulated with other data sources (Campbell et al., 2013; Khandkar, 2009).

Applying the conjecture mapping analytical tool

As we moved forward in our analysis, we applied the conjecture mapping analytical tool to the full data set. This analytical move was in line with our research question about the challenges instructors reported facing with respect to providing sustained debugging support. At this stage of our analysis, we applied the method of interrater agreement, or negotiated

agreement, in which "two or more coders are able to reconcile through discussion whatever coding discrepancies they may have for the same unit of text" (Campbell et al., 2013). Given the exploratory nature of our research question, we followed the precedent of relying on interrater agreement "where generating new insights is the primary concern" (p. 306) *before* implementing a check for interrater reliability [see Campbell et al.'s (2013) paper for additional detail]. Consistent with approaches to DBR that attend to *contingencies* in the deployment of a design (Cobb et al., 2003), we narrowed the analytical tool to five reflection practices that each represented a type of pushback on design: (a) *impromptu mediating processes*[2], which consisted of events in class that instructors used to explain how or why their designs did or did not lead to specific short-term outcomes for specific (groups of) students; (b) *new features* denoted any actions, events, or outcomes instructors implemented in their teaching that were not already on our original conjecture maps; (c) *predictions and hopes* marked events that instructors described either wanting to happen in their class or predicting would happen if certain conditions occurred; (d) *confusion and uncertainty* were moments in which instructors expressed doubt about a learning design they had used or described uncertainty in understanding what they observed in class; and (e) *backward connections* were descriptions of how a short-term outcome an instructor observed during class led them to make a particular choice about what teaching approach to try next (moving from right to left on the conjecture map).

Using a spreadsheet, Author 1 worked through responses from conjecture map 1 (focused on journal-based reflections), inserting direct quotes from instructors into one of five reflective practice categories. Once all of the instructors' comments were categorized for conjecture map 1, each researcher then separately examined the data, met and discussed any discrepancies in data categorization, refined definitions, and reached a common understanding before repeating this process for conjecture maps 2 and 3.

Once we reached consensus on the categorization of responses within each conjecture map, we worked to summarize what we considered to be central themes for each of the five reflective practices. For our first attempt, each research team member worked alone to aggregate the data into overall themes—collapsing and separating themes as necessary—with respect to a single reflection practice on a single conjecture map. After each researcher completed this analysis, we reconvened to describe, compare, and revise our aggregated themes. We then repeated this process with each reflective practice for each conjecture map.

Next, we examined themes across conjecture maps. We carried out this final analysis over the course of several months. Each researcher worked separately to compare, aggregate, and refine overall themes for each reflective practice across conjecture maps. For example, in the mediating processes category across maps, several of the themes that instructors voiced pertained to norms, expectations, and consistency; we grouped these themes together into one overarching category (for an example, see Table 2). Following this analysis, we worked together over several sessions to present, debate, and ultimately aggregate themes across reflective practices, removing duplicate themes and grouping related themes with new overarching labels. We repeated this process over two successive rounds. This final pass amounted to the themes that we present in the "Results" section.

[2] Mediating processes are a core feature of Sandoval's (2014) original conception of conjecture mapping. In our use of the construct, we are describing instructors' proposed additional mediating processes that were not noted in their original conjecture maps. These were impromptu mediating processes that instructors noticed when tensions or unexpected events arose in the classroom.

Table 2 Example of organized emerging themes of instructor push back for a reflection practices

Emerging themes for new features reflection practice	
Theme	Reflection
Relationship between goals and processes	• Have students pick a certain strategy alongside their goal
	• Reinforce goals throughout the day
Relevance of debugging	• Students develop understanding of the importance of debugging rather than just how to do it
Storytelling about teachers, peers, and experts failing and enacting the debugging process	• Student participation increases when instructors use personal examples of bugs, instructors then incorporated that into lessons, especially to model process
	• Students explain what they have done in the past when talking to each other
	• Sharing historical or fictional debugging stories with students such as Grace Hopper's actual bug stuck in a computer, emphasizing trials and experimentation, especially before goal setting period for students, and trying to relate bugs with "real world analogies", in service of getting comfortable with bugs
	• Instructors putting bugs into lessons (including code written on the whiteboard) and having students predict whether it will work and solve them as a comprehension check
	• If students catch a bug in instructor code, use it as the next days debugging story
	• Students tried to remember what they had done in a previous class to help them
	• Class made "dank" debugging memes to engage students and help students in reflection on their journals to foster class community
	• Debugging hashtags on post-its to reflect on coding session at end of class
Structure of the debugging process	*Impromptu/Specific/Individual (Student-driven)*
	• Students developing personal examples of bugs they solved
	• Instructor tries to understand a student's own process when working with them
	Formulaic/Generic/Legitimate (Instructor-endorsed)
	• Students beginning to use common strategies
	• Have the whole class focus on the same goal
	• Debugging processes still seem foreign to some students
Norms and expectations	• Through repetition students are taking up terminology around debugging
	• Instructors emphasize reflection for students
	• Stories help to normalize struggle and bugs and encourages students to develop positive attitudes about debugging
Scaffolded debugging	• Instructors tried "telling the story" of the bug (but might forget to enforce in class)
	• Students would try to remember what they had done in previous class to help other students
Student agency in debugging	• [Not present in new features reflections]
The emotional experience of debugging	• [Not present in new features reflections]

Approach to triangulation and interrater agreement

Despite that our analysis consisted primarily of instructor's written reflections, in our above-described process, we triangulated claims by looking across the providers of the data (different instructors), focal points of reflection (different conjecture maps), and analytical lenses (the distinct components of our analytical tool), searching in particular for points of convergence (Tracy, 2010). That is, by using our analytic tool to focus separately on each reflection practice (e.g., referencing, unpacking, backward connections, etc.) within each conjecture map prior to aggregating instructors' statements into themes, we helped ensure that our findings would not be narrowly relevant to one particular design, one particular instructor, or one particular type of noticing in the classroom. Our analysis process involved sustained, collaborative work to create the analytical tool, independent application of the tool to the data set by each researcher in our team, and aggregation of themes that stretched across debugging designs, multiple instructors, and multiple facets of the analytical tool.

We also adhered to the qualitative research recommendation of conducting an interrater reliability check after thorough interrater agreement efforts (Campbell et al., 2013). Given that we had already worked through the full data set of instructors' written reflections on their conjecture maps, to conduct an interrater reliability and data triangulation check, we examined a separate data source: the earlier mentioned interviews with instructors after the summer CS workshops. Each research team member coded separate hour-long interviews, and transcribed and categorized any moment in which the instructors both referenced one of the eight themes and noted friction around that theme (e.g., discussing how coordinating goals and processes was hard to enact in the classroom). Each researcher then created a blank version of the coded data and asked the other researcher to match themes to the data. Both researchers fully converged on their coding of the data for all but two themes. Our subsequent discussion about this discrepancy in this coding process led us to clarify a key distinction between student agency and instructor scaffolding. Namely, we clarified that our notion of student agency focused on any ways that instructors viewed students as driving and/or shaping their work, and we clarified that scaffolding referred to instructors' efforts to actively support their students. Afterward, we repeated the above interrater reliability process focused specifically on student agency and scaffolding. For this round, we coded an interview that we had not previously watched, and found that independent raters converged on their categorization of the data for 90% of the instructors' statements. Following this agreement check, we then revisited our coded data to look for any evidence of this discrepancy. Altogether, this process both demonstrated a high amount of interrater reliability when examining our themes in a new data set, and demonstrated that our findings triangulated with a new data source. That is, through this process, we confirmed that all eight themes were discussed by instructors in all of the post summer workshop interviews.

Reflections on positionality in our DBR partnership

Centering our identities as CS education researchers, we wish to start this positionality statement by noting a few of the reasons why we approached this design-based research with a strong commitment to prioritizing teacher expertise. First, we recognized that there was a strong leaning in debugging education research toward student processes and outcomes over and above instructors' experiences, and we wanted to mitigate this trend.

Second, we had worked together with our non-profit partner to conceive of the grant proposal that funded this work, and in deep acknowledgement that they were the ones serving students and families on a daily basis, we sought at all stages of our work to have the non-profit and its teachers lead the design of the learning environment. Third, we had been closely reading and learning from scholarship that foregrounds participatory DBR partnerships that de-center educational research priorities (e.g., Bang & Vossoughi, 2016), and this further reaffirmed our commitment in this study to understanding teachers' in-classroom experiences. In addition, heeding Folkes (2022) call to reflect on how positionality changes and evolves throughout a study, we note that throughout our summer work, we repeatedly signaled to instructors our open mind when it came to designing and evaluating the efficacy of our team's debugging pedagogy. We implemented this concretely by encouraging instructors throughout the study to adapt their debugging pedagogy designs and push back on predictions in their conjecture maps in ways authentic to their unfolding experiences in the classroom. In addition, as educational researchers who had not taught large classrooms of CS students, we felt that it was essential to honor instructors' voices in our data collection and analysis efforts.

Moreover, we spent our time in all three summers of data collection with instructors in the classroom when they were teaching. Author 2 was present in the classroom in the focal summer workshops taking field notes, occasionally working one-on-one with students, and reflecting with instructors; Author 1 followed this same approach in the DBR partnership's second summer of data collection. Author 2 also served as a teacher and researcher in the second summer of data collection, co-leading a classroom resembling the ones under consideration in the present paper, and following the same protocol of collecting video data and reflecting on conjecture maps. In talking to instructors every day, taking field notes in the classroom, and teaching in the classroom ourselves, we experienced and observed the daily work, the difficulties, and the improvised adaptations of teaching debugging, and we appeal to the significance of the findings in this paper at least in part to honor the work that instructors invest in CS classroom teaching. We also wish to foreground Tracy's (2010) emphasis that "relationally ethical investigators engage in reciprocity with participants and do not co-opt others just to get a 'great story" (p. 847). Our work continued with these instructors for 3 years, and as we detail in the "Discussion" section, we revised our designs in alignment with instructors' reflections on what was difficult about teaching debugging. Finally, we recognize that there are many facets of privilege to our positions as learning scientists who are not often asked to shoulder the responsibility of labor, creative energy, and constant adaptation that practicing teachers experience daily. We sought in this study to honor as best as possible the work of the instructors, what they experienced as difficult, what they adapted as they taught, and what they deemed ineffective in the classroom, as one stream of influence on upcoming CS education professional development and debugging designs.

Results

We articulate eight themes from instructors' reflections on their approach to providing sustained debugging support to students. These themes mark the heterogeneous ways that instructors pushed back on the design choices and rationales from their conjecture maps, articulated by envisioning new mediating processes, backward connections, new features, confusion/uncertainty, and/or hopes and predictions about how the designs might

eventually work. Below, we present each theme in its own section; quotes referenced in these sections are direct statements from instructors. The "Discussion" section below then explores the broader implications of possible interconnections between these themes.

Theme #1: relationships between debugging goals and processes

This theme describes instructors' thoughts about how students' personal goals with respect to exploring and using debugging (written in journals at the start of class) related to the debugging processes students enacted when working to fix broken code. Instructors described actively planning a wide range of efforts to "integrate the debugging goals better into the actual practice of coding." These teaching moves included referencing goals during debugging and at the end of class; asking students to reflect on their goals; asking students to pursue their debugging goals before soliciting help; asking students to describe what debugging goals they set last time and whether they had pursued any of those goals already; and asking students to set a goal for the next bug based on their experience with bugs earlier that day. Instructors understood goal setting as critical to mediating students' understanding of debugging and the terminology surrounding it.

Redesign efforts focused on interleaving debugging goals and processes stemmed from instructors having noticed that students' goals set at the start of class could fade away over time, remaining in students' closed journals without students reflecting on them during coding. In one instructor's words, "often students will not really think about or reflect on their goal for that day, even when they are reminded by us of it. They just write it down in the beginning, but then there isn't much serious reflection and it's hard to see if they're gaining from the time we spend on it." When working with the goal-setting design, instructors responded to friction derived from a disconnect between debugging goals and processes. Instructors unpacked teaching moves that brought the two in dialogue by stitching goals and practices together before, during, and after coding, instead of passively expecting that a written goal at the start of a lesson would have an automatic, cascading impact on coding practice.

Theme #2: relevance of debugging goals and strategies

Instructors' comments on this topic focused on whether debugging was viewed as relevant to a given circumstance, and why students cared about growing their debugging practices. On the first, instructors noted that on some days students either coded very little or did not come across the types of bugs that would have been relevant to the debugging goals they had set for themselves. In response, instructors predicted that creating additional challenges that engaged students in debugging, including tailored coding situations focused on specific types of bugs, would help make specific debugging goals and strategies more relevant (e.g., "Perhaps create some engaging activities focusing on certain strategies so students can explore them in isolated situations, rather than be overwhelmed by an abundance of choices"). Alternatively, an instructor ranked some bugs as more relevant than others, for example, by offering as a debugging strategy the correct syntax so that the broken syntax bugs could be fixed quickly, freeing up students to focus on logic bugs. In addition, instructors observed that the relationship between a specific bug and the content of the day's coding lesson mediated whether students enacted or ignored certain debugging strategies.

In a similar vein, instructors sought to understand why students had chosen to focus on particular types of growth in their debugging practices. Instructors described nudging

students to explain why they chose a particular debugging strategy to focus on as their daily goal (e.g., "After the students answer the prompt, I ask the students to share their debugging goals and explain why they chose a particular debugging strategy"). One instructor's proposed redesign would have asked students not only to pick a goal in their journals, but also to write a rationale for why they had selected that goal. In focusing on why a debugging strategy or goal was meaningful to students' daily coding, instructors predicted that students would develop an understanding of the importance of debugging in the coding process. In short, instructors persistently called attention to friction with respect to the meaningfulness of debugging goals and strategies. Why a particular debugging approach mattered to students, and whether that debugging strategy aligned with the problem facing the student, were core concerns of instructors.

Theme #3: storytelling about teachers, peers, and experts failing and enacting the debugging process

This theme addresses instructors' comments about the value of students gaining access to how other individuals, especially experts, navigate moments of failure (e.g., Lin-Siegler et al., 2016). Instructors utilized their own impromptu mistakes (whether during live coding or outside of coding) as a way to emphasize how integral failure and debugging were to routine activities (e.g., "In yesterday's lesson, the students pointed out a bug in one of my slides. I used that moment for the following day's debugging story-of-the-day to model the debugging process step by step of the students"). They noted that student participation increased when instructors focused on their own, personal examples of bugs during the class, and predicted that instructors transparently talking about their own debugging goals might be useful for students to see. Several instructors then incorporated this into their lessons by purposely putting bugs into their coding examples during lessons in order to model their debugging process ("while writing the program, we purposely put in a bug and…we went through the debugging process together as a class"). Instructors stated that this redesign led to students discussing and explaining the bugs that they encountered when working with a smaller group or partner.

Along with sharing accounts of their own bugs, instructors shared historical and fictional debugging stories with students, such as the story of the actual insect that Grace Hopper discovered inside a computer. These stories emphasized trials and experimentation with buggy code, using "real world analogies" in service of normalizing encounters with bugs in code. This evolved throughout the duration of the class so that when an instructor incorporated a bug example in their lesson, if a student was able to point it out, instructors would use that as the next day's debugging story for the class. For example, an instructor noted that "in a previous lesson, when I was teaching the students functions, the students noticed that there was a logic error in my sample code. I used that moment as our debugging story of the day the following day and used the bug in my code to model each step of the debugging process to fix the error." One instructional team even extended this storytelling process by generating classroom memes about students' journal reflections on debugging. In short, instructors perceived storytelling about failure as a valuable window into the failure response process but noticed that focusing only on stories external to their school community was inadequate. Instead, they described surfacing their own failures, linking storytelling about failure to bugs students had noticed, and honoring in micro stories (or memes) student's own experiences.

Theme #4: scaffolded debugging

In this section, we cover instructors' reflections on how experts scaffold students' approaches to debugging. By scaffolding, we mean the introduction of supports during debugging that would eventually fade away in students' subsequent debugging approaches. Unique to this category, instructors positioned their thoughts about scaffolding as outcomes they hoped would emerge and as new design features, clearly envisioning future-oriented revisions. Toward this end, instructors noted that taking time to model debugging approaches, especially with increased frequency, and reflecting on their process outwardly with difficult bugs might ultimately lead to debugging becoming easier and more comfortable for students. In addition, instructors noted the difficulty of reflecting after the fact on one's previous debugging process, and proposed remedying this by modeling the process for students to "point out the highlights" in students' approaches. Similarly, prompting students in specific ways—to describe the history of the problem (i.e. "every bug has a story") and using specific debugging language—and then gradually fading out those prompts while observing carefully served as a way to see if students enacted those strategies on their own.

Many scaffolds not originally reflected in the conjecture maps emerged as well. Instructors positioned student thinking as a key indicator of how they should scaffold, moving the "listening" aspect of debugging instruction to the start of the process by resisting the allure of advancing to the answer too quickly and instead asking more open-ended questions about debugging during one-on-one sessions. They additionally introduced the notion of helping students plan their code as a key scaffold to set up successful debugging. In order to nudge students out of only using the strategies comfortable to them, instructors prompted students during debugging to explore new strategies. For whole-class activities, instructors described summarizing why a particular bug will make a program stop working or probing whether a bug the class encountered was an error of logic or syntax. As debugging became commonplace in the classroom, instructors noted that students began to help each other point out bugs or recommend strategies that helped them solve a different bug. Coding resources such as packets that included helpful example syntax, called "zenes," were also used to compare students' bugs with some common examples of bugs in code. In all, instructors pushed back on the notion that debugging growth would take place without systematic methods to scaffold students' debugging, whether that entailed surfacing their own debugging thinking, modeling approaches, nudging exploration of new strategies, or drawing attention to physical resources in the classroom.

Theme #5: norms and expectations

This theme lays out instructors' views on the value of building up classroom norms and expectations around debugging. Instructors used words such as "through repetition," "normalize," "reinforce," "make it an expectation," "remind," and "emphasize." These norms included setting goals in journals, telling debugging stories, and giving daily reminders about debugging strategies when coding. They noted that increasing the frequency of time spent debugging may help students take up debugging terminology, but would not be enough. They described a need to embed debugging terminology in their day-to-day modeling of debugging, and having students think about and narrate their process when working with others. Instructors noted that providing and modeling examples of debugging would legitimize debugging as a practice that expert coders also consistently carry out.

Setting goals and telling stories based on debugging were described as a norm that helped to position failure as a necessary part of coding for students. Instructors talked about how goals centered around individual strategies helped students become more familiar with them as time went on. Instructors placed an expectation that students remember their debugging goal throughout the day's lesson. Storytelling "helped to normalize struggle and bugs and encouraged students to develop positive attitudes about debugging" (our emphasis). Instructors reinforced debugging as a practice by making it an expectation that students "walk me through the debugging process, especially narrating your process and articulating what specific resources/strategies they're using." Instructors also sought to make resources such as the debugging journal and reflections on post-it notes "more of a norm" so that students could reflect on debugging experiences through multiple activities. Overall, instructors commonly returned to a refrain that new debugging practices would not simply emerge in the classroom when introduced, but rather, that instructors would need to work toward a debugging culture through explicit classroom norms.

Theme #6: student agency in debugging

This section covers instructors' thoughts surrounding student agency during debugging. This referred to moments or interactions where students marked a sense of ownership over their own coding or debugging strategies. Instructors operationalized agency along a number of dimensions of student behavior observed in the classroom: using debugging vocabulary and tools without prompting; proposing solutions to their bugs and pinpointing the location of new bugs; writing down their debugging process in their daily journal entries; taking it upon themselves to remind instructors to distribute certain class resources, such as "coding zenes"; and expressing they "wanted to figure the answer out themselves." To foster some of these forms of agency, instructors noted wanting to create more time for students to code their personal projects, a setting in which they often observed strong student voice and involvement, one of the key outcomes for their conjecture maps. Other instructors noticed that student agency, with respect to both confidence and knowledge, would emerge through social interaction with instructors. That is, instructors would respond to a student's request for help and then observe that the student already had a firm handle on that bug.

However, instructors also consistently addressed challenges they experienced when fostering students' agency in the debugging process. One instructor frankly noted early in a workshop that he had yet to see students engaged in the debugging process. In response to these kinds of observations, instructors reflected openly about their hopes and predictions toward this end. They aimed to ask more open-ended questions about debugging to "get at student's thinking and perspective." They described wanting to slow down during one-on-one interactions to "let students explain their bug" to them without rushing straight to the answer for students. Instructors also noted wanting to praise students' use of technical language, and promote the specific names of strategies students use in debugging. Overall, instructors noted a range of ways that students expressed agency during debugging, but also recognized that deliberate attention to dynamics around student agency could extend their impact.

Theme #7: structure of the debugging process

As instructors reflected on their approach to agency (described in the section above), they commonly discussed the structure of the debugging process. Though this topic is closely coupled to agency, it was also so prevalent in instructors' reflections and specifically focused on the steps of debugging that it warranted separating it out as a unique theme. In all, this category lays out the tensions that instructors encountered around a "one-size-fits all" approach to debugging. They noted two effective, but opposing, ways to structure debugging: instructor-endorsed versus student-driven approaches. For instructor-endorsed structures, instructors described finding it necessary to stress the importance of a formalized debugging process. These instructor-endorsed debugging strategies (e.g. explaining your code to a rubber duck) were treated at times as "legitimate" debugging practices for students. Instructors wanted to ensure that students understood a focal "strategy of the day" for the class so that the strategies could "become more automatic." By having the class as a whole focus on the same debugging strategy, instructors noted that students could then use these common strategies to solve bugs and reflect on the efficacy of these common strategies together. This structure of the debugging process was a common thread across conjecture maps and focused on the class as a whole homing in on specific debugging strategies.

On the other hand, instructors highlighted student-driven approaches to debugging, which some described as "impromptu" and connected to individual students when specific bugs arose in their code. In these instances, instructors noted that students developed personal examples of how they solved bugs, and instructors would work to understand students' distinct processes of debugging to be able to help them. The debugging processes here were often one-on-one interactions between an instructor and student, and instructors described working to understand the student's process by asking open-ended questions as opposed to "directing them on what to do." There was an emphasis on being able to "narrate your process" and articulate the strategies being used. Instructors leaning towards the student-driven structure of debugging described it as "more organic and productive for the students to debug their own way." It allowed instructors to gauge what aspect of the debugging process they might then emphasize with the student. Overall, our analysis documented a persistent tension between instructor-driven and student-driven approaches to debugging.

Theme #8: the emotional experience of debugging

This section outlines the importance that instructors placed on the heterogeneous emotional states of students and themselves during the debugging process (e.g., Dahn & DeLiema, 2020; Dahn et al., 2020; Kinnunen & Simon, 2010). Instructors noted factoring in students' emotions when deciding whether to push students to keep struggling with a bug or more directly guide students through the process. Instructors intentionally noted if students were feeling frustrated, excited, or even neutral with bugs found in their code. For example, difficult problems could leave students outwardly frustrated and wanting to give up. Instructors wrote that students would get frustrated when they "weren't feeling a sense of progress," and get disappointed by their bugs. Instructors described improvisationally changing their teaching approach as students reacted emotionally to the debugging process. For example, if students became frustrated after a series of prompts from the instructor, an instructor might walk students back to a previous step in the curriculum and model the process for

them again. Instructors also described feeling their own frustration when working with students who were not making progress on debugging.

Likewise, instructors wrote about how their positive attitude and framing of bugs mediated the efficacy of their debugging instruction. They noted that students would respond positively when they "felt a sense of progress" during debugging and would show excitement when their code started to work. Instructors sometimes started class with stories that helped normalize struggle and encouraged students to go after more challenging bugs. Instructors overall described their students as "in touch with their emotions" while debugging. To help promote positive orientations to bugs, instructors discussed trying to prepare students for what it "feels like when encountering a bug while coding." It was important for them to frame alternative ways to share the emotional experience of debugging that "felt like less of a chore." For example, they used bug "hashtags" created by students for class humor and what an instructor described as "casual and natural talk" in their classrooms. Instructors reflected that when their class had positive emotional reactions to debugging, it allowed them to create challenges that placed students in harder positions, and students would push themselves through the challenge. To summarize, instructors latched on to the critical role that students' and their own emotions played in the efficacy of debugging teaching and learning. Taking note of the positive, negative, and even neutral emotions that emerged in their classrooms was part of the process that shaped how instructors adapted their debugging instruction from moment to moment.

Discussion

This study engaged with instructors' day-to-day experiences providing sustained debugging support to students in a summer coding workshop. The first contribution of this paper was describing the development of an analytical tool for categorizing instructors' fine-grained reflections on conjecture maps. This analytical tool captures a number of ways that instructors retrospectively analyzed predictions from their own conjecture maps (Sandoval, 2014), namely what instructors considered to be shortcomings in the first versions of the maps. Serving as a boundary object that helped stitch together instructors' disparate viewpoints (Penuel et al., 2015; Star & Griesemer, 1989; Suchman, 1994), the conjecture map provided a reference point for instructors' noticing of the friction they encountered in their teaching, in particular what they predicted/hoped would happen in their classrooms, what was confusing/uncertain, how they responded to short-term outcomes with new instructional moves, and what factors needed to be in place to mediate the effectiveness of the designs. We explicitly valued continuity between instructors' predictions at the outset of the study and their reflections throughout the study, and thus used the conjecture map as a persistent artifact instructors could use to gauge their changing thinking over time. Indeed, collaborative efforts to design and reflect on conjecture maps with teachers are increasingly common in DBR (David et al., 2019; Leonard et al., 2017; Penuel et al., 2016), and which is reflective of a broader movement toward participatory design research (Bang & Vossoughi, 2016; Jurow et al., 2016; Vakil et al., 2016; Zavala, 2016). Instructors' expectations and intentions were embedded in the features highlighted in their conjecture maps (including how items in the map were phrased), which provided a personal point of departure from which to describe how designs were then implemented in the classroom and how students responded. The conjecture maps served as a lens through which instructors could

critique instructional design decisions, raise tensions that arose in their day-to-day work, and articulate how they responded and what they considered along the way.

Our conjecture mapping analytic tool similarly focused on the contingencies that arose in instructional work (Cobb et al., 2003) and made possible learning from these points of tension (Engeström, 2001). When we examined the full data set, the analytical tool required us to acknowledge a wide range of ways that instructors pushed back on instructional design decisions, before we aggregated them into broader themes. We would argue that this analytical tool can play a part in participatory design-based research efforts by nudging researchers to anticipate a number of ways that instructors might comment on improvable facets of instructional designs. Our intention in highlighting this analytical tool as a contribution of this paper is that it may give other research-practice teams a means to study teachers' participation in the construction and retrospective analysis of collaborative design conjectures. In our case, this meant examining instructors' reflections to understand where our learning designs were falling short, and what these shortcomings might mean more broadly for supporting students through failure, topics we consider next.

Applying the analytical tool in the context of debugging

The second contribution of this paper was our application of this analytic tool in the context of a DBR study. Because research on teachers' experiences supporting students with debugging is quite rare in the literature (e.g., Michaeli & Romeike, 2019), it is valuable to know that these themes arise in the context of CS instruction around debugging. Though many of these themes align with findings in the broader educational literature—such as research on sociodisciplinary norms (e.g., Yackel & Cobb, 1996)—our work here shows that these themes are concerns of CS educators when they are supporting debugging, and furthermore maps these themes to specific tensions in the computational practice of debugging. For example, against the backdrop of recent scholarship in CS education resisting the unquestioned appropriation of disciplinary norms in CS education spaces (Fong et al., 2020; Ryoo et al., 2020; Philip & Sengupta, 2021; Tissenbaum et al., 2021), our work here invites CS educators and researchers to consider whether the friction instructors experienced around setting norms that "position failure as necessary" and as disciplinarily common derived in part from the promotion of a value that simply clashed with students' expectations and priorities in the classroom. Granted that the norm is worth setting, our work here at the least invites CS educators to move beyond studies of debugging strategies and tools and think carefully about the classroom setting in which these tools and techniques arise. In other words, the domain specificity of these themes is a key contribution of the paper. In addition, because our work was longitudinal, these themes draw attention to practical realities of sustained efforts to teach debugging that should be valuable both for future research and professional development.

We argue that the themes uncovered through the application of our conjecture mapping analytic framework to our data provide a range of considerations for educators and researchers interested in designing and/or studying CS education learning spaces that supports students' debugging. We articulate them in the following way:

- Establish connections between students' goals with respect to failure and their enacted practices. This might mean understanding the goals students formulate when it comes to failure and ensuring that the debugging opportunities, they encounter provide authentic contexts for them to explore these goals and strategies.

- When sharing stories about expert approaches to failure, consider also modeling one's own failures and honoring the students' own stories/experiences.
- Strike a balance between understanding/valuing students' own approaches to debugging and introducing known, effective strategies and processes for debugging.
- Introduce norms around debugging by setting clear expectations and embedding debugging activities and terminology throughout the learning experience.
- Scaffold students' approaches to failure by surfacing expert debugging thinking, modeling approaches to debugging, nudging exploration of new debugging strategies, and/or drawing attention to debugging resources in the classroom.
- Affirm and understand students' agency with debugging by asking open-ended questions, slowing down at impasses, and giving students credit for their debugging approaches.
- Calibrate instructional approaches to the moment-by-moment emotions students experience during debugging.

Our approach of positioning instructors' day-to-day reflections as a key data source in DBR reflects recent work in CS education research (Fields et al., 2018; Michaeli & Romeike, 2019; Ryoo, 2019), and marks a substantial departure from prior CS education research that has tended to ignore the human experience of teaching and learning coding (see Sengupta et al., 2021, for a critique of this tradition). The themes we unpacked in this paper introduce a new challenge to researchers of failure and learning by inviting research not solely about one or two of these themes, but also about how instructors and students navigate failure at the intersection of these multiple facets and across sustained periods of time in the classroom. To be more specific, the CS education literature contains research papers that separately cover the themes we found in this paper. For example, there is some evidence that students who use provided debugging strategies effectively resolve code problems, but do not always utilize these strategies in practice (Ko et al., 2019). The findings from our paper raise questions about whether instructors similarly experience a persistent tension between guiding students toward structured debugging processes and attending to and supporting students' preferred approaches to fixing problems. Separately, inviting students to design their own buggy artifacts creates authentic debugging contexts and generates high interest and confidence (Fields et al., 2020). More broadly, research on K-12 CS instructors' experiences corroborate some of the themes we uncovered, including tensions supporting students' individualized problem-solving strategies in one-on-one sessions (Yadav et al., 2016). However, the power of these papers in providing a specific focus on a particular facet of debugging eschews the heterogeneous difficulties teachers experience over long time horizons as they provide sustained debugging support to students. For this reason, our longitudinal design and our data collection approach centering teachers' experiences make the themes that we uncovered in this paper particularly valuable to educational leaders and researchers interested in better supporting CS teachers—and by extension their students—in the classroom.

Teachers in this setting were grappling in their instructional approach with all of these facets of debugging pedagogy. In response to this finding, we emphasize that research and teaching that explicitly grapple with *connections* between these themes would more responsively and actionably address the instructional realities of providing sustained debugging support over multiple learning sessions and toward heterogeneous ends in naturalistic learning spaces. A number of vital questions emerge at the intersection of these themes: When should each be foregrounded in instruction, who should decide which of these themes will most anchor a particular debugging culture in a classroom, and what

additional tensions (beyond the ones instructors noted here) arise when multiple of these themes are simultaneously pursued? We acknowledge that these questions and considerations may be especially relevant to learning environments that share attributes of the one we studied, such as low teacher to student ratios, synchronous work around students' code, pedagogical goals that center students' autonomy with debugging, sustained debugging support stretching across multiple weeks of instruction, and instructors who are relatively new to teaching computer science. We also believe that these themes will be valuable to researchers outside of computer science. For instance, outward talk "attending to or interpreting failures" (e.g., stating aloud that failure has arisen and explaining the cause of the failure) is rare in some maker settings (Simpson et al., 2019). In reflecting on the implications of their study, Simpson et al. (2019) ask if "this practice [should] be modeled by educators" (p. 9) to provide an example to students of how to talk openly about failure and its causes. We would point out that the teachers in our study noted similar pedagogical possibilities, and we might go further to wonder about how norm setting, emotion, debugging process structures, etc. might also guide makerspace pedagogy surrounding moments of failure. Lastly, in formulating these themes, we do not mean to imply that they can be pursued in a simple way. We instead view these themes as realistic constraints and goals that instructors encounter when providing sustained debugging support in a naturalistic learning environment, and in this way, might be viewed as important considerations for educators in other contexts working toward designing a robust approach to supporting students across repeated encounters with failure.

Revisions to learning designs following the summer CS workshops

Following the summer of data collection reported in this paper, we adhered to the DBR practice of iterating on a number of our learning designs through pilot work during weekend workshops and then with a new stable version the following summer. These revisions were responsive to a number of the themes uncovered in the current analysis. Revisions included instructors holding small group huddles tailored to a few students instead of whole-class discussion; instructors fostering argumentation among peers during debugging; students having leeway to select coding challenge difficulty levels and move at an independent pace; daily efforts by instructors to honor students' debugging experiences with stories students wrote and then connected to a tree in the courtyard of the workshop space; and arts-based reflections on debugging emotions, strategies, and goals that were incorporated before, during, and after coding instead of taking place in a separate classroom. By aiming to honor students' personal stories with debugging and emotion (Themes #3 and #8), tying students' reflections more closely to their coding practice (Theme #1–2), giving students authority to move at their own pace (Theme #6), and setting a precedent for continuous peer dialogue around debugging (Theme #5), these design revisions were in alignment with the themes uncovered in the present paper. For example, after bringing students' arts-based reflections on emotion into the coding classroom (instead of housing them in a separate art space), we created opportunities to normalize the heterogeneous experience of emotion in the classroom (Themes #5 and #8), and have subsequently documented how these designs "opened possibilities for the learners to observe, understand, and critically examine the integration between problem solving, emotion, and identity in their programming experience" (Dahn & DeLiema, 2020; p. 362). Moreover, from a methodological perspective, we extended our learnings from the first summer. In particular, our DBR team continued to reflect on our designs via conjecture maps, but we provided

printouts of the maps so that instructors could mark them up directly (as we had done in our analysis for this paper), and we made more space for collaborative reflection on the conjecture maps. In all, considerably more work is needed to refine these pedagogical and methodological designs and test their efficacy in naturalistic learning settings. For example, in DBR efforts extending from this paper's findings, we are investigating specific themes, such as the structure of the debugging process (Theme #7), and exploring through collaborative conjecture mapping and collaborative design with middle school CS educators how alternative debugging processes that recognize open-ended pathways and non-linear strategies would ameliorate the tensions around structure recognized in the present paper (e.g., Wilson Vasquez et al., accepted). We hope that the themes reported here, and the analytical tool we shared, will be similarly generative for educational researchers in and outside of CS.

Limitations

We highlight here a number of limitations of this study. The first limitation we note is that our analysis did not address facets of power within the process of creating the instructional designs and within the process of developing the conjecture maps. Educational researchers moving forward with similar approaches might consider what other participation frameworks would give teachers more authority in the process of developing design conjectures. Our DBR team aimed to give teachers agency to articulate their observations, elaborations, and critiques of the team's design conjectures; agency to pilot and refine the designs in the weekend workshops leading up to this summer's study; and a voice in contributing to redesigns piloted the following winter and spring. However, a robust body of scholarship troubles the assumption that researchers have neutral partnerships with educators (e.g., Bang and Vossoughi, 2016; Vakil et al., 2016). Acknowledging that we fell short of attending to these dynamics within our DBR team, we call for future conjecture mapping research with instructors that attend to power in the process of foregrounding particular values, voices, and rationales in design work (e.g., Lee et al., 2022). These considerations are central to the construction of design conjectures as much as they are central to the analysis of design conjectures. For example, had we moved faster with the development of our conjecture mapping analytic tool, our analysis of the instructors' reflections on their design conjectures could have involved member checking to provide an additional angle of triangulation on the eight themes we aggregated from the data. Future work could also more deeply attend to how instructors' professional development experiences shape the kinds of contributions they make to design conjectures. In our summer professional development period, instructors were managing quite a lot (e.g., learning about the curriculum, preparing their classrooms) when they were formulating design conjectures; this might have unintentionally proved to be a less nurturing setting for design planning than we had envisioned.

A second limitation is that this project took place in the informal learning context of a summer coding camp. While parts of the program have found their way into "formal" schooling spaces through our non-profit collaborator, the uniqueness of the summer space—small class sizes, students interested in programming from the outset, no state-required learning objectives, instructors with limited experience teaching—make it likely that different themes might emerge in different settings. A third limitation of this study is that the analysis emphasized instructors' written reflections over other sources of data, such as collected video data, including how students experienced debugging around each of these themes. Our prior work in this context documented the development of students'

confidence and awareness in using debugging strategies (DeLiema et al., 2020), and we chose here to focus explicitly on instructors' experiences and *their* judgments of effectiveness. While we did analyze a set of instructor interviews for the purposes of triangulation, future analyses could aim to triangulate this methodological approach with data related to student experience, including video data or interviews. Lastly, the analysis did not aim to quantify the frequency of instructors' concerns around a given theme, nor did this analysis track how/whether redesign efforts alleviated any of the tensions instructors experienced in the classroom. Given these considerations, we cannot make strong claims about how to integrate these themes in practice, nor can we note which themes are most important and/or most representative of what happens in the classroom.

Conclusion

Design-based research in which instructors are positioned as core contributors to both the construction and retrospective analysis of design conjectures can provide a unique window into the efficacy of instructional design decisions. In particular, paying attention to the friction that instructors notice in their day-to-day teaching through the boundary object of conjecture maps can throw light on where a learning community might dedicate time to revising its teaching and learning practices. Critically, this DBR work is situated in the context of CS education where these themes, while present in the literature, are valuable to recognize for CS educators. There are many teachers who are newly moving toward teaching CS (Haduong & Brennan, 2019; Menekse, 2015; Sullivan & Gresalfi, 2020; Warner et al., 2019)—similarly to the teachers in our study—and we hope that the findings here can inform professional development as much as they can inform concrete debugging pedagogies. When applied to a learning environment in which instructors and researchers were focused on the quality of debugging support, we were able to examine the instructional reality of navigating heterogeneous facets of supporting students across moments of failure and challenge the field to envision professional development and learning designs around debugging that straddle these multiple facets of instructional support.

Funding This work was supported by the National Science Foundation under Grant Nos. 1612770, 1607742, and 1612660.

Data availability Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest No conflict of interests by the authors.

References

Adair, J. K., & Kurban, F. (2019). Video-cued ethnographic data collection as a tool toward participant voice. *Anthropology & Education Quarterly*, *50*(3), 313–332.

Bang, M., & Vossoughi, S. (2016). Participatory design research and educational justice: Studying learning and relations within social change making. *Cognition and Instruction*, *34*(3), 173–193.

Bakker, A. (2018). *Design research in education: A practical guide for early career researchers*. Routledge.

Bismack, A., Arias, A. M., Davis, E. A., & Palincsar, A. S. (2015). Examining student work for evidence of teacher uptake of educative curriculum materials. *Journal of Research in Science Teaching, 52*(6), 816–846.

Bland, L. C., Hjalmarson, M. A., Nelson, J. K., & Samaras, A. S. (2017). Applying conjecture mapping as a design-based research method to examine the design and implementation of a teaching development project for STEM faculty. In *ASEE Annual Conference proceedings*.

Boud, K., Koegh, R., & Walker, D. (1985). Promoting reflection in learning: A model. In K. Boud, R. Koegh, & D. Walker (Eds.), *Reflection: Turning experience in learning* (pp. 18–40). Kogan.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada (Vol. 1, p. 25).

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141–178.

Brown, J., & Crippen, K. (2016). Designing for culturally responsive science education through professional development. *International Journal of Science Education, 38*(30), 470–492.

Campbell, J. L., Quincy, C., Osserman, J., & Pedersen, O. K. (2013). Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. *Sociological Methods & Research, 42*(3), 294–320.

Çimer, A., Çimer, S. O., & Vekli, G. S. (2013). How does reflection help teachers to become effective teachers. *International Journal of Educational Research*, *1*(4), 133–149.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, *32*(1), 9–13.

Collins, A. (1992). *Toward a design science of education. New directions in educational technology* (pp. 15–22). Springer.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical & methodological issues. *The Journal of the Learning Sciences, 13*(1), 15–42.

Danish, J. A. (2014). Applying an activity theory lens to designing instruction for learning about the structure, behavior, and function of a honeybee system. *Journal of the Learning Sciences*, *23*(2), 100–148.

David, S. S., Pacheco, M. B., & Jiménez, R. T. (2019). Designing translingual pedagogies: Exploring pedagogical translation through a classroom teaching experiment. *Cognition and Instruction*, *37*(2), 252–275.

Day, C. (1999). *Developing teachers: The challenges of lifelong learning*. Falmer Press.

Dahn, M., & DeLiema, D. (2020). Dynamics of emotion, problem solving, and identity: Portraits of three girl coders. *Computer Science Education*, *30*(3), 362–389.

Dahn, M., Deliema, D., & Enyedy, N. (2020). Art as a point of departure for understanding student experience in learning to code. *Teachers College Record, 122*(8), 1–42.

DeLiema, D., Bye, J., & Marupudi, V. (2021). Programming instructors and students' active (and partial) debugging: Deviation noticing, causal modeling, and intervening. Paper presented at *the annual conference of the American Educational Research Association*, Virtual Meeting, Virtual Annual Meeting.

DeLiema, D., Dahn, M., Flood, V. J., Asuncion, A., Abrahamson, D., Enyedy, N., & Steen, F. F. (2020). Debugging as a context for collaborative reflection on critical thinking and emotion. In E. Manolo (Ed.), *Deeper learning, communicative competence, and critical thinking: Innovative, research-based strategies for development in 21st century classrooms* (pp. 209–228). Routledge.

DeLiema, D., Kwon, Y., Chisholm, A., Williams, I., Dahn, M., Flood, V., Abrahamson, D., & Steen, F. (2022). A multi-dimensional framework for documenting students' heterogeneous experiences with programming bugs. *Cognition & Instruction*. https://doi.org/10.1080/07370008.2022.2118279.

Derry, S. J., Pea, R. D., Barron, B., Engle, R. A., Erickson, F., Goldman, R., Hall, R., Koschmann, T., Lemke, J. L., Sherin, M. G., & Sherin, B. L. (2010). Conducting video research in the learning sciences: Guidance on selection, analysis, technology, and ethics. *The Journal of the Learning Sciences*, *19*(1), 3–53.

DesPortes, K., & DiSalvo, B. (2019). Trials and tribulations of novices working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 219–227).

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. DC Heath and Company.

Dickes, A. C., Farris, A. V., & Sengupta, P. (2020). Sociomathematical norms for integrating coding and modeling with elementary science: A dialogical approach. *Journal of Science Education and Technology*, *29*, 35–52.

diSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *The Journal of the Learning Sciences, 13*(1), 77–103.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *The Journal of the Learning Sciences, 11*(1), 105–121.
Endres, A. (1975). An analysis of errors and their causes in system programs. *IEEE Transactions on Software Engineering*. https://doi.org/10.1109/TSE.1975.6312834
Engeström, Y. (2001). Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of Education and Work*, *14*(1), 133–156.
Engeström, Y. (2011). From design experiments to formative interventions. *Theory & Psychology*, *21*(5), 598–628.
Farrell, C. C., Penuel, W. R., Coburn, C., Daniel, J., & Steup, L. (2021). *Research-practice partnerships in education: The state of the field*. William T. Grant Foundation.
Fields, D. A., & Kafai, Y. B. (2020). Debugging by Design: Students' Reflections on Designing Buggy E-Textile Projects. *In Proceedings of Constructionism 2020*.
Fields, D. A., Kafai, Y., Nakajima, T., Goode, J., & Margolis, J. (2018). Putting making into high school computer science classrooms: Promoting equity in teaching and learning with electronic textiles in exploring computer science. *Equity & Excellence in Education*, *51*(1), 21–35.
Flood, V. J., DeLiema, D., Harrer, B. W., & Abrahamson, D. (2018). Enskilment in the digital age: The interactional work of learning to debug. In J. Kay & R. Luckin (Eds.), *"Rethinking learning in the digital age: Making the Learning Sciences count," Proceedings of the 13th International Conference of the Learning Sciences* (Vol. 3, pp. 1405–1406). London: International Society of the Learning Sciences.
Folkes, L. (2022). Moving beyond 'shopping list' positionality: Using kitchen table reflexivity and in/visible tools to develop reflexive qualitative research. *Qualitative Research*. https://doi.org/10.1177/14687941221098922
Fong, M., Aalst, O. W. V., Flood, V., & DeLiema, D. (2020). When features become bugs: Stance-taking around refactoring in a coding classroom. In Y. Kafai (Chair), Turning bugs into learning opportunities: Understanding debugging processes, perspectives, and pedagogies. In M. Gresalfi, M. & I. S. Horn (Eds.), *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences* (ICLS) 2020, Volume 2 (pp. 374–381). Nashville, TN: International Society of the Learning Sciences.
Glaser, G. B. (1965). The constant comparative method of qualitative analysis. *Social Problems*, *12*(4), 436–445.
Gomoll, A., Hmelo-Silver, C. E., & Šabanović, S. (2022). Co-constructing professional vision: Teacher and researcher learning in co-design. *Cognition and Instruction*, *40*(1), 7–26.
Goodwin, C. (1994). Professional vision. *American Anthropologist*, *96*(3), 606–633.
Goodwin, C. (2018). *Co-operative action*. Cambridge University Press.
Gutiérrez, K. D. (2008). Developing a sociocritical literacy in the third space. *Reading Research Quarterly, 43*(2), 148–164.
Gutiérrez, K. D., & Vossoughi, S. (2010). Lifting off the ground to Return Anew: Mediated praxis, transformative learning, and social design experiments. *Journal of Teacher Education, 61*(1), 100–117.
Gutiérrez, K. D., Engeström, Y., & Sannino, A. (2016). Expanding educational research and interventionist methodologies. *Cognition and Instruction*, *34*(3), 275–284.
Gutiérrez, K., & Jurow, S. (2016). Social design experiments: Toward equity by design. *Journal of the Learning Sciences, 25*(4), 565–598.
Haduong, P., & Brennan, K. (2019). Helping K–12 Teachers Get Unstuck with Scratch: The Design of an Online Professional Learning Experience. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 1095–1101).
Hall, R., & Stevens, R. (2015). Interaction analysis approaches to knowledge in use. *Knowledge and Interaction* (pp. 88–124). Routledge.
Hassenfeld, Z. R., & Bers, M. U. (2020). Debugging the writing process: Lessons from a comparison of students' coding and writing Practices. *The Reading Teacher*, *73*(6), 735–746.
Heikkilä, M., & Mannila, L. (2018). Debugging in programming as a multimodal practice in early childhood education settings. *Multimodal Technologies and Interaction*, *2*(3), 42.
Hennessy Elliott, C., Gendreau Chakarov, A., Bush, J. B., Nixon, J., & Recker, M. (2023). Toward a debugging pedagogy: Helping students learn to get unstuck with physical computing systems. *Information and Learning Sciences*. https://doi.org/10.1108/ILS-03-2022-0051
Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and correcting Java programming errors for introductory computer science students. *ACM SIGCSE Bulletin*, *35*(1), 153–156.
Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39–103.

Jurow, A. S., Teeters, L., Shea, M., & Van Steenis, E. (2016). Extending the consequentiality of "invisible work" in the food justice movement. *Cognition and Instruction*, *34*(3), 210–221.
Kafai, Y. (2006). Constructionism. In K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 35–46). Cambridge University Press.
Kali, Y., Sagy, O., Kuflik, T., Senior Member, I. E. E. E., Mogilevsky, O., & Maayan-Fanar, E. (2015). Harnessing technology for promoting undergraduate art education: A novel model that streamlines learning between classroom, museum, and home. *IEEE Transactions on Learning Technologies, 8*(1), 5–17.
Kapur, M. (2008). Productive failure. *Cognition and Instruction, 26*(3), 379–424.
Khandkar, S. H. (2009). Open coding. *University of Calgary, 23*, 2009.
Kelly, A. E. (2004). Design Research in Education: Yes, but is it methodological? *Journal of the Learning Sciences*, *13*(1), 115–128.
Kim, C., Yuan, J., Vasconcelos, L., Shin, M., & Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, *46*(5), 767–787.
Kinnunen, P., & Simon, B. (2010). Experiencing programming assignments in CS1: the emotional toll. In *Proceedings of the Sixth international workshop on Computing education research* (pp. 77–86).
Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20*(3), 362–404.
Ko, A. J., & Myers, B. A. (2005). A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages & Computing, 16*(1–2), 41–84.
Ko, A. J., & Myers, B. A. (2009). Finding causes of program output with the Java Whyline. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1569–1578).
Ko, A. J., LaToza, T. D., Hull, S., Ko, E. A., Kwok, W., Quichocho, J., Akkaraju, H., & Pandit, R. (2019). Teaching explicit programming strategies to adolescents. In Proceedings of the *50th ACM Technical Symposium on Computer Science Education* (pp. 469–475).
Lampert, M., Franke, M. L., Kazemi, E., Ghousseini, H., Turrou, A. C., Beasley, H., Cunard, A., & Crowe, K. (2013). Keeping it complex: Using rehearsals to support novice teacher learning of ambitious teaching. *Journal of teacher education*, *64*(3), 226–243.
Land, S. M., & Zimmerman, H. T. (2015). Socio-technical dimensions of an outdoor mobile learning environment: A three-phase design-based research investigation. *Education Tech Research Development, 63*(2), 229–255.
Lazar, T., Sadikov, A., & Bratko, I. (2017). Rewrite rules for debugging student programs in programming tutors. *IEEE Transactions on Learning Technologies*, *11*(4), 429–440.
Lee, U. S., DeLiema, D., & Gomez, K. (2022). Equity conjectures: A methodological tool for centering social change in learning and design. *Cognition & Instruction*, *40*(1), 77–99.
Lee, V. R., & Dubovi, I. (2020). At home with data: Family engagements with data involved in type 1 diabetes management. *Journal of the learning sciences*, *29*(1), 11–31.
Lee, V. R., Recker, M., & Phillips, A. L. (2018). Conjecture Mapping the Library: Iterative Refinements Toward Supporting Maker Learning Activities in Small Community Spaces. In J. Kay & R. Luckin (Eds.), *Rethinking Learning in the Digital Age: Making the Learning Sciences Count*, 13th International Conference of the Learning Sciences (ICLS) 2018 (Vol. 1, pp. 320–327). London, UK:ISLS.
Lee, V. C., Yu, Y. T., Tang, C. M., Wong, T. L., & Poon, C. K. (2018). ViDA: A virtual debugging advisor for supporting learning in computer programming courses. *Journal of Computer Assisted Learning*, *34*(3), 243–258.
Leonard, S. N., Belling, S., Morris, A., & Reynolds, E. (2017). Playing with rusty nails:'Conceptual tinkering' for 'next' practice. *EDeR: Educational Design Research*. https://doi.org/10.15460/eder.1.1.1027
Levin, J. R., & O'Donnell, A. M. (1999). What to do about educational research's credibility gaps? *Issues in Education, 5*(2), 177–229.
Lewis, C. M., & Gregg, C. (2016). How Do You Teach Debugging? Resources and Strategies for Better Student Debugging. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 706–706).
Liberman, N., Kolikant, Y. B., & Beeri, C. (2012). Regressed experts" as a new state in teachers' professional development: Lessons from Computer Science teachers' adjustments to substantial changes in the curriculum. *Computer Science Education*, *22*(3), 257–283.
Lin-Siegler, X., Ahn, J. N., Chen, J., Fang, F. F. A., & Luna-Lucero, M. (2016). Even Einstein struggled: Effects of learning about great scientists' struggles on high school students' motivation to learn science. *Journal of Educational Psychology*, *108*(3), 314.
Lowe, T. (2019). Debugging: The key to unlocking the mind of a novice programmer?. In *2019 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). IEEE.

Matuk, C., Gerard, L., Lim-Breitbart, J., & Linn, M. (2016). Gathering requirements for teacher tools: Strategies for empowering teachers through co-design. *Journal of Science Teacher Education*, *27*(1), 79–110.

McCauley, R., Manaris, B., Mazzone, M., & Bares, W. (2010). Computing in the arts: A model curriculum. In *Proceedings of the 45th ACM technical symposium on Computer science education* (SIGCSE '14). Association for Computing Machinery, New York, NY, USA, 451–456.

Menekse, M. (2015). Computer science teacher professional development in the United States: A review of studies published between 2004 and 2014. *Computer Science Education*, *25*(4), 325–350.

Michaeli, T., & Romeike, R. (2019). Current Status and Perspectives of Debugging in the K12 Classroom: A Qualitative Study. In *2019 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1030–1038). IEEE.

Miljanovic, M. A., & Bradbury, J. S. (2017). Robobug: a serious game for learning debugging techniques. In *Proceedings of the 2017 acm conference on international computing education research* (pp. 93–100).

Nasir, N. I. S., & Cooks, J. (2009). Becoming a hurdler: How learning settings afford identities. *Anthropology & Education Quarterly*, *40*(1), 41–61.

Nasir, N. S., & Vakil, S. (2017). STEM-focused academies in urban schools: Tensions and possibilities. *Journal of the Learning Sciences*, *26*(3), 376–406.

Palinscar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and instruction, 1*(2), 117–175.

Papert, S. (1980). "Mindstorms" Children. *Computers and powerful ideas*.

Penuel, W. R., Allen, A. R., Coburn, C. E., & Farrell, C. (2015). Conceptualizing research–practice partnerships as joint work at boundaries. *Journal of Education for Students Placed at Risk (JESPAR)*, *20*(1–2), 182–197.

Penuel, W. R., Bell, P., Bevan, B., Buffington, P., & Falk, J. (2016). Enhancing use of learning sciences research in planning for and supporting educational change: Leveraging and building social networks. *Journal of Educational Change*, *17*(2), 251–278.

Penuel, W. R., Roschelle, J., & Shechtman, N. (2007). Designing formative assessment software with teachers: An analysis of the co-design process. *Research and Practice in Technology Enhanced Learning, 2*(1), 51–74.

Philip, T. M., Bang, M., & Jackson, K. (2018). Articulating the "how," the "for what," the "for whom," and the "with whom" in concert: A call to broaden the benchmarks of our scholarship. *Cognition and Instruction*, *36*(2), 83–88.

Philip, T. M., & Sengupta, P. (2021). Theories of learning as theories of society: A contrapuntal approach to expanding disciplinary authenticity in computing. *Journal of the Learning Sciences*, *30*(2), 330–349.

Phillips, D. C., & Dolle, J. R. (2006). From Plato to Brown and beyond: Theory, practice, and the promise of design experiments. In L. Verschaffel, F. Dochy, M. Boekaerts, & S. Vosniadou (Eds.), *Instructional psychology: Past, present and future trends: Sixteen essays in honour of Erik DeCorte* (pp. 277–293). Elsevier.

Prather, J., Pettit, R., Becker, B. A., Denny, P., Loksa, D., Peters, A., Albrecht, Z., & Masci, K. (2019). First things first: Providing metacognitive scaffolding for interpreting problem prompts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 531–537).

Prediger, S., Gravemeijer, K., & Confrey, J. (2015). Design research with a focus on learning processes: An overview on achievements and challenges. *Zdm Mathematics Education*, *47*(6), 877–891.

Rich, K. M., Strickland, C., Binkowski, T. A., & Franklin, D. (2019). A K-8 debugging learning trajectory derived from research literature. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 745–751).

Ripley, G. D., & Druseikis, F. C. (1978). A statistical analysis of syntax errors. *Computer Languages*, *3*(4), 227–240.

Ryoo, J. J. (2019). Pedagogy that supports computer science for all. *ACM Transactions on Computing Education (TOCE)*, *19*(4), 1–23.

Ryoo, J. J., Tanksley, T., Estrada, C., & Margolis, J. (2020). Take space, make space: How students use computer science to disrupt and resist marginalization in schools. *Computer Science Education*, *30*(3), 337–361.

Saleh, A., Hmelo-Silver, C. E., Glazewski, K. D., Mott, B., Chen, Y., Rowe, J. P., & Lester, J. C. (2019). Collaborative inquiry play: A design case to frame integration of collaborative problem solving with story-centric games. *Information and Learning Sciences*, *120*(9), 547–666.

Sanders, E. B. N., & Stappers, J., P (2008). Co-creation and the new landscapes of design. *Co-design*, *4*(1), 5–18.

Sandoval, W. A. (2004). Developing learning theory by refining conjectures embodied in educational designs. *Educational Psychologist, 39*(4), 213–223.
Sandoval, W. A. (2014). Conjecture mapping: An approach to systematic educational design research. *Journal of the Learning Sciences, 23*(1), 18–36.
Santo, R., Vogel, S., Ryoo, J., Denner, J., Belgrave, C., Moriss, A., & Tirado, A. (2020). Who Has a Seat at the Table in CSed? Rethinking Equity Through the Lens of Decision-making and Power in Computer Science Education Initiatives. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 329–330.
Schön, D. A. (1987). *Educating the reflective practitioner: Toward a new design for teaching and learning in the professions*. Jossey-Bass.
Schön, D. A. (1992). Designing as reflective conversation with the materials of a design situation. *Knowledge-Based Systems, 5*(1), 3–14.
Sengupta, P., Dickes, A., & Farris, A. V. (2021). *Voicing code in STEM: A dialogical imagination*. MIT Press.
Severance, S., Penuel, W. R., Sumner, T., & Leary, H. (2016). Organizing for teacher agency in curricular co-design. *Journal of the Learning Sciences*, *25*(4), 531–564.
Shavelson, R. J., Phillips, D. C., Towne, L., & Feuer, M. J. (2003). On the science of education design studies. *Educational researcher*, *32*(1), 25–28.
Shaw, M. S., Fields, D. A., & Kafai, Y. B. (2020). Leveraging local resources and contexts for inclusive computer science classrooms: Reflections from experienced high school teachers implementing electronic textiles. *Computer Science Education*, *30*(3), 313–336.
Sherin, M., & van Es, E. (2005). Using video to support teachers' ability to notice classroom interactions. *Journal of technology and teacher education*, *13*(3), 475–491.
Silvis, D., Clarke-Midura, J., Shumway, J. F., Lee, V. R., & Mullen, S. (2022). Children caring for robots: Expanding computational thinking frameworks to include a technological ethic of care. *International Journal of Child-Computer Interaction*. https://doi.org/10.1016/j.ijcci.2022.100491
Simpson, A., Anderson, A., & Maltese, A. V. (2019). Caught on camera: Youth and educators' noticing of and responding to failure within making contexts. *Journal of Science Education and Technology, 28*, 480–492.
Spohrer, J. C., Soloway, E., & Pope, E. (1985). Where the bugs are. *ACM SIGCHI Bulletin*, *16*(4), 47–53.
Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39. *Social Studies of Science, 19*(3), 387–420.
Stromholt, S., & Bell, P. (2018). Designing for expansive science learning and identification across settings. *Cultural Studies of Science Education*, *13*(4), 1015–1047.
Suchman, L. (1994). Working relations of technology production and use. *Computer supported cooperative work*, *2*(1–2), 21–39.
Sullivan, F. R., & Gresalfi, M. S. (2020). Beyond inclusion: The imperative of criticality in CS education. *Computer Science Education*, *30*(3), 249–253.
The Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for Educational Inquiry. *Educational Researcher*, *32*(1), 5–8.
Tissenbaum, M., Weintrop, D., Holbert, N., & Clegg, T. (2021). The case for alternative endpoints in computing education. *British Journal of Educational Technology*, *52*(3), 1164–1177.
Tobar-Muñoz, H., Baldiris, S., & Fabregat, R. (2017). Augmented reality game-based learning: Enriching students' experience during reading comprehension activities. *Journal of Educational Computing Research*, *55*(7), 901–936.
Tobin, J., Hsueh, Y., & Karasawa, M. (2009). *Preschool in three cultures revisited: China, Japan, and the United States*. University of Chicago Press.
Tracy, S. J. (2010). Qualitative quality: Eight "big-tent" criteria for excellent qualitative research. *Qualitative Inquiry*, *16*(10), 837–851.
Tucker-Raymond, E., Puttick, G., Cassidy, M., Harteveld, C., & Troiano, G. M. (2019). I broke your game!": Critique among middle schoolers designing computer games about climate change. *International Journal of STEM Education*, *6*(1), 41–57.
Vakil, S., McKinney de Royston, M., Suad Nasir, N. I., & Kirshner, B. (2016). Rethinking race and power in design-based research: Reflections from the field. *Cognition and Instruction*, *34*(3), 194–209.
van den Akker, J. (1999). Principles and methods of development research. In J. van den Akker, R. M. Branch, K. Gustafson, N. Nieveen, & T. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 1–14). Springer.

van den Akker, J. (2013). Curricular development research as a specimen of educational design research. In T. Plomp & N. Nieveen (Eds.), *Educational design research* (pp. 53–70). Netherlands Institute for Curriculum Development (SLO).

van Es, E. A., & Sherin, M. G. (2010). The influence of video clubs on teachers' thinking and practice. *Journal of mathematics teacher Education*, *13*(2), 155–176.

van Es, E. A., Cashen, M., Barnhart, T., & Auger, A. (2017). Learning to notice mathematics instruction: Using video to develop preservice teachers' vision of ambitious pedagogy. *Cognition and Instruction*, *35*(3), 165–187.

Vasconcelos, L., Arslan-Ari, I., & Ari, F. (2020). Early childhood preservice teachers' debugging block-based programs: An eye tracking study. *Journal of Childhood Education & Society*, *1*(1), 63–77.

Vea, T. (2020). The learning of emotion in/as sociocultural practice: The case of animal rights activism. *Journal of the Learning Sciences*, *29*(3), 311–346.

Vedder-Weiss, D., Ehrenfeld, N., Ram-Menashe, M., & Pollak, I. (2018). Productive framing of pedagogical failure: How teacher framings can facilitate or impede learning from problems of practice. *Thinking Skills and Creativity*, *30*, 31–41.

Vygotsky, L. S. (1978). *Mind in society: Development of higher psychological processes*. Harvard university press.

Warner, J. R., Fletcher, C. L., Torbey, R., & Garbrecht, L. S. (2019). Increasing capacity for computer science education in rural areas through a large-scale collective impact model. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1157–1163). Minneapolis, MN.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Syslo, M. M. (2017). Computer science in K-12 school curricula of the 21st century: Why, what and when? *Education and Information Technologies*, *22*(2), 445–468.

Wilkerson, M. H. (2017). Teachers, students, and after-school professionals as designers of digital tools for learning. *Participatory design for learning: Perspectives from practice and research*, pp. 1–13.

Wilson Vazquez, A., DeLiema, D., Goeke, M., & Bye, J. (accepted). Debugging debugging instruction: A research-practice partnership in K-8 computer science education. In upcoming proceedings of the *International Conference of the Learning Sciences* (ICLS) 2023. Montreal: Canada. International Society of the Learning Sciences.

Wozniak, H. (2015). Conjecture mapping to optimize the educational design research process. *Australasian Journal of Educational Technology, 31*(5), 597–612.

Yackel, E., & Cobb, P. (1996). Sociomathematical norms, argumentation, and autonomy in mathematics. *Journal for Research in Mathematics Education, 27*(4), 458–477.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235–254.

Zavala, M. (2016). Design, participation, and social change: What design in grassroots spaces can teach learning scientists. *Cognition and Instruction*, *34*(3), 236–249.

Zeller, A. (2009). *Why programs fail: A guide to systematic debugging*. Morgan Kauffman.