Characterizing Teacher Support of Debugging with Physical Computing: Debugging Pedagogies in Practice

COLIN HENNESSY ELLIOTT, Drexel University School of Education, Philadelphia, PA, USA
JESSIE NIXON, Weber State University, Ogden, UT, USA
ALEXANDRA GENDRAU CHAKAROV, JEFFREY B. BUSH, and MICHAEL J. SCHNEIDER, University of Colorado, Boulder, CO, USA
MIMI RECKER, Utah State University, Logan, UT, USA

Objectives. Physical computing systems are increasingly being integrated into secondary school science and STEM instruction, yet little is known about how teachers, especially those with little background and experience in computing, help students during the inevitable debugging moments that arise. In this article, we describe a framework, comprising two dimensions, for characterizing how teachers support students as they debug a physical computing system called the Data Sensor Hub (DASH). The DASH enables students to program sensors to measure, analyze, and visualize data as they engage in science inquiry activities.

Participants. Five secondary school teachers implemented an inquiry-oriented instructional unit designed to introduce students to working with the DASH as a tool for scientific inquiry.

Study Method. Findings drew on video analysis of the teachers' classroom implementations of the unit. A review of the data corpus led to the selection of 23 moments where the teachers supported an individual or small groups of students engaged in debugging. These moments were analyzed using a grounded perspective based on Interaction Analysis to characterize the teachers' varied interactional approaches.

Findings. Our analysis revealed how teachers' moves during debugging moments fell along two dimensions. The first dimension characterizes teachers' positioning during the debugging interactions, ranging from a positioning for teacher understanding to a positioning for student understanding of the bug. The second dimension characterizes the inquiry orientation of the teachers' questions and guidance, ranging from focusing on the debugging process to focusing on the product—or fixing the bug. Further, teachers' moves often fell along different points on these dimensions given nuances in the instructional context.

Conclusions. The framework offers a first step toward characterizing teachers' debugging pedagogy as they support students during debugging moments. It also calls attention to how teachers do not necessarily need to be programming experts to effectively help students learn independent and generalizable debugging strategies. Further, it illustrates the variety of expertise that teachers can bring to debugging moments to support students learning to debug. Finally, the framework provides implications for the design of professional learning and supports for teachers as they increasingly are asked to support students in computing—and debugging—activities across a range of disciplines.

This research was funded by the National Science Foundation (Award No. 1742053, No. 2019805, and No. 1742046) and the James S. McDonnell Foundation. The opinions expressed are those of the authors and do not represent the views of the funding organizations.

Authors' Contact Information: Colin Hennessy Elliott (corresponding author), Drexel University School of Education, Philadelphia, PA, USA; e-mail: ch3457@drexel.edu; Jessie Nixon, Weber State University, Ogden, UT, USA; e-mail: Jessie.Nixon1@weber.edu; Alexandra Gendrau Chakarov, Colorado School of Mines, Golden, CO, USA; e-mail: agc@colorado.edu; Jeffrey B. Bush, University of Colorado, Boulder, CO, USA; e-mail: Jeffrey.Bush@colorado.edu; Michael J. Schneider, University of Colorado, Boulder, CO, USA; e-mail: Michael.J.Schneider@colorado.edu; Mimi Recker, Utah State University, Logan, UT, USA; e-mail: mimi.recker@usu.edu.

ACM 1946-6226/2024/12-ART48
https://doi.org/10.1145/3677612

CCS Concepts: • **Social and professional topics → K-12 education**; *Computer science education*; *Computing literacy*; *Student assessment*;

Additional Key Words and Phrases: Debugging, Physical Computing Systems, K-12 Computer Science Integration

ACM Reference format:
Colin Hennessy Elliott, Jessie Nixon, Alexandra Gendrau Chakarov, Jeffrey B. Bush, Michael J. Schneider, and Mimi Recker. 2024. Characterizing Teacher Support of Debugging with Physical Computing: Debugging Pedagogies in Practice. *ACM Trans. Comput. Educ.* 24, 4, Article 48 (December 2024), 28 pages.
https://doi.org/10.1145/3677612

1 Introduction

While the range of education technologies to support K-12 students in learning computing has grown rapidly (e.g., [37, 52]), the study of teaching and learning how to debug in the context of these tools has remained mostly focused on the experiences of the learner. For example, McCauley et al. [43] examined the depth and breadth of early debugging literature with four major questions: "Why do bugs occur, what types of bugs occur, what is the debugging process, and how can we improve the learning and teaching of debugging?" (p. 68). In response, a growing body of research has continued to examine the practices of learning to debug from a variety of approaches including: (1) developing a bug taxonomy [1], (2) conceptualizing debugging as an activity of productive failure [30, 33], (3) exploring the conceptual change required to become a better debugger [48], (4) articulating the knowledge required to learn debugging [38], (5) explicitly teaching debugging strategies [35], and (6) examining the emotional process of debugging [8].

Recently, a line of research has expanded debugging research beyond pure software programming to analyze how students approach debugging while using **physical computing systems (PCS)**, for example, Arduino-based systems [13] and electronic-textiles [16, 27]. This article builds on this line of work through integrating computational practices into middle school science (or STEM) classes using a specific sensor-based PCSs and accompanying inquiry-oriented curriculum. This sensor-based PCS consists of a programmable microcontroller which receives data from sensors as inputs, processes the inputted data, and then takes action through connected actuators (e.g., moving servos, lighting up LEDs, vibrating buzzers). These "hybrid projects" can provide opportunities for deeper learning [27] as students explore the interconnected areas of hardware and software [13, 28, 31, 39].

When embedded into science inquiry activities, sensor-based PCSs can be tools for developing students' epistemic agency [46] where they take ownership of the knowledge-building process in scientific inquiry, both individually and collectively [2]. Therefore, physical computing with sensors offers an opportunity to integrate computing into secondary school science and STEM classrooms and presents a unique challenge for teachers to support students debugging and troubleshooting as they program and build their own technologies for conducting science inquiry.

Whether teachers are new to programming or not, integrating PCSs into secondary school science or STEM curriculum can create important moments of uncertainty for the teacher where they may be unsure of how to support students. When students ask for help with their PCS that is not working, the teacher may not be able to diagnose the underlying problem quickly or know exactly how to support the student in finding and fixing the problem themselves or both. In these moments, the teacher must make rapid decisions to help the student become "unstuck." How teachers approach these moments of uncertainty can be influenced by a variety of factors—from class length and size to the physical structures of the classroom—as well as their own prior experiences with computing.

Prior scholarship has researched teachers learning to debug PCSs (e.g., [34]), how teachers support students during classroom discussions around debugging (e.g., [6, 13, 24]), and how teachers' personal reflections can shed light on how they support their students' debugging [56]; little research currently attends to the varied moves teachers make in-the-moment to support their students in becoming "unstuck" when working with PCS. Taking up DeLiema and colleagues [10] charge to "more inclusively and responsibly frame debugging as an open-ended process" (p. 8), this article thus adds to the growing field of research attending to the interactions between students and teachers during moments of debugging (e.g., [12, 18]), by examining teachers' embodied moves as they support students' debugging of a sensor-based PCS [19]. We analyze instances where teachers interact with small groups of students working on making their individual PCS work. The approaches teachers take to these interactions, or debugging moments as we call them, has implications for the debugging-specific pedagogies, or what we are calling debugging pedagogies.

Our study is guided by the following research question: *What moves do teachers make to help support students in developing debugging practices?* Using a sociocultural theoretical frame, we analyze debugging moments that occurred during five secondary teachers' classroom implementations of an inquiry-oriented instructional unit designed to introduce students to working with a sensor-based PCS called the **Data Sensor Hub (or DASH)** [20] as a tool for scientific inquiry. Our grounded analysis led to a characterization of teachers' moves along two dimensions as a first step toward developing a theory of teachers' debugging pedagogy. These two dimensions describe how teachers' moves are oriented (1) to the debugging process (process/product dimension) and (2) to their own or their students' understanding (understanding dimension).

2 Prior Work on Learning to Debug

Debugging and learning to debug is central for students learning computational practices in STEM classrooms. Papert [48] explained that "[e]rrors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it" (p. 114). Newcomers to programming often struggle with learning to debug their systems [16] because, as scholars have pointed out, debugging skills differ from the skills of programming (e.g., [1]). When students are "stuck" after encountering a bug [22], they often stop and wait for the instructor to locate the bug [50], get support from a peer, or both [27].

To understand why students struggle with debugging their projects, research from the 1980s focused on categorizing the kinds of bugs students experience and the strategies they employ for debugging [43]. Many concluded that students often do not form a "big picture," which Vessey [60] refers to as a "system view," of the system they are debugging. Students who lack a clear picture of how the subsystems work together to form the entire system demonstrated a less systematic debugging process. Perkins et al. [50] referred to students who had a tendency for sporadic changes (that can create more bugs) as "Tinkerers." Unlike "Movers," who take a more methodical debugging approach and appear to work toward narrowing down the potential cause of a system's buggy behavior, Tinkerers will keep making changes, jumping from one failed hypothesis to the next. While Tinkerers can, and often do, fix the bug they may not fully understand how they fixed the bug, limiting their skill development for recognizing and rectifying future bugs. In contrast, "Stoppers" hit a proverbial wall and stop testing bug hypotheses, waiting for someone else to come and fix the bug for (or with) them. In other words, these students become "stuck" [22]. Recognizing where students are during the debugging process—whether they are a Mover or a Stopper—may help teachers assess how to best help students in the moment. As another example, Pea et al. [49] suggest employing research-like techniques such as clinical interviews to understand students' perspectives on certain repetitive bugs. Further, DeLiema et al. [10] have recently articulated a conceptualization of debugging situations in educational settings where teachers and students

negotiate different "debugging pathways" which offer possible solutions to making the program work. In other words, moments of debugging are heterogenous opportunities for learning and development [11].

Debugging exposes students' emerging learning needs, creating important moments for support and helping teachers focus on targeted areas of struggle [22, 49]. Teachers play an important role in these moments and can help students develop the cognitive, social, and emotional capacities necessary to work through complex problems [40]. For example, DeLiema et al. [11] offer debugging activities as an important context for students and teachers to engage in communication about critical thinking. They suggest that learning to debug should include reflection and storytelling which promote communication about the learning process itself.

PCSs further complicate the debugging process [5] because they include connected hardware and software that have exclusive or connected issues [13] making the "systems view" [60] even harder for students to grasp as they navigate tendencies to Tinker, Move, and Stop. From another perspective, physical computing components create more possibilities for debugging pathways, as evidenced in our previous case study of one teacher's work [24]. Using the context of high schoolers working with electronic-textiles, Jayathirtha et al. [26] articulate this as the "difficulty of coordinating between various components of physical computing systems … the physical artifact, representations on paper, and the onscreen programming environment" (p. 1053). To support students in locating bugs in both hardware and software, Fields et al. [15] provided learners buggy e-textile systems where they had to find and fix planted bugs (e.g., a short circuit (hardware) or compiler error (software)). They found that the physicality of the projects made thinking about and prioritizing the order of problem solving to be an important part of debugging. The PCS, like the one students used in this article, included sensors to collect information about the system's surrounding environment, adding an additional component to coordinate.

3 Conceptual Framework: A Sociocultural Approach to Understanding Debugging Pedagogies in Practice

3.1 Debugging as Recognition and Reconciliation of Inconsistencies

Ko and Myers [36] describe the debugging process beginning at the identification of an issue, or failure, in the system. Thus, there is a pedagogical element to helping students notice an issue [10], in addition to supporting them to fix issues when they are "stuck." Using a sensor-based PCS, students encounter emergent material resistance [42] that requires tinkering with components and programs that measure and process the data streams when outputs do not match expectations (e.g., a classroom temperature reading is 200°F). This entails recognizing what data and models mean in relation to their lived experiences that inform their expectations for outputs [4]. We start from the assumption that bugs in computational systems come to existence when they are interactionally identified [12] through recognition of inconsistencies between the system's actual output and the expectation of what the system's output should be. Debugging, therefore, is an open-ended process of developing, evaluating, and negotiating possible "debugging pathways" [10]. Building on Suchman [59], Flood et al. [18] identify the process of debugging as a situated inquiry where debuggers fix issues caused by bugs by constantly asking questions about the system and remaking their paths of inquiry. Such a situated inquiry cannot be explicitly taught but must be learned through developing skills.

Conceptualizing debugging as situated inquiry informs our theoretical approach to analysis grounded in sociocultural theories of learning and development [61]. This approach helps us explain how developing debugging skills and practices is a social and cultural process of developing relations. The environments, activities, and communities where learners are learning to debug

shape the contours and possibilities of how they can learn. From this perspective, each debugging interaction is different; learners are learning approaches and skills of inquiry rather than a particular repeatable practice.

This theoretical base supports our conceptualization of teachers' interactional offerings—or what we're calling teacher moves—as resources for students learning to debug as part of joint meaning making of possible solution pathways. Further, it supports our conceptualization of teachers' approaches as indicative of their debugging pedagogies, or perspectives and practices informed by views of how students learn to debug.

3.2 The Teacher's Role in Supporting Students during Debugging

How learners learn to debug has been studied from an array of learning theories [32]. Yet, as DeLiema et al. (Dave Toce) point out, existing research on debugging teaching and learning has mostly focused on the actions of students while debugging, and often assumes a singular problem and solution pathway. As we have pointed out in previous work [24, 47], theorizing, analyzing, and understanding the role of the teacher in supporting students to debug their systems is understudied. Often this translates to the assumption that teachers must learn to be expert debuggers before they can support students and misses the nuances of teachers' in-the-moment facilitation in supporting others learning to debug [24].

Recent research highlights the variety of frictions teachers experience while supporting students' debugging. These include helping students understand the relevance of debugging, helping them draw connections between their goals and the processes they take, navigating the various ways to structure debugging, and negotiating the various way to scaffold debugging [10]. Understanding how to best support students' debugging is further complicated as most debugging skills are taught on demand [44], often requiring teachers to make in the moment decisions that require structured improvisation [57]. Further, there is an emerging scholarship developing on what kinds of practices and knowledges it takes to make culturally responsive debugging.

3.3 Debugging Pedagogies

Scholars who have focused on teachers' activities in supporting students learning to debug in various informal learning environments have argued for a need to explore such practices at interactional scales [10]. Flood et al. [18], for example, demonstrate the ways that instructors can effectively support a learner's development of debugging skills through key guidance on strategies that are authentic to the debugging experience, similar to guidance needed in learning more physical activities using a power tool. As described above, there is emerging scholarship (e.g., [7, 11, 21]) that describes the value of explicit reflection on critical thinking and emotions while learning to debug. Therefore, facilitating authentic guidance during learners' debugging practice and associated reflective practice is grounded in what we call *debugging pedagogy*.

Debugging pedagogy is the undergirding perspectives and motivations that shape how teachers—and other facilitators—make decisions about how to approach supporting students during the debugging process. These perspectives can help develop what Papert [48] calls a "debugging philosophy" where "errors benefit us…" because they lead to developing an understanding of the system. In some instances, a teacher sharing something with the whole class about how to debug a PCS is shaped by perspectives about how students can learn to debug. Similarly, a perspective informs how a teacher celebrates students' persistence in trying to get their PCS to work [62] or encourages students to see the systems' failures as learning opportunities [30].

3.4 Teacher Moves as the Unit of Analysis

In this study, we apply a sociocultural theory [61] to understand the teachers' role as relational supports who adapt to the learner's trajectory, social positioning, and developing relationship to the materials. We look to learn from teachers' practices undergirded by their debugging pedagogies. To this end, we use "teacher moves" as the unit of analysis, which are the actions teachers make in-the-moment when interacting with students who are debugging. These moves include a coordination of talk, gestures, and physical movements around the student and their system. The analysis starts from the assumption that a significant portion of teachers' actions to support students learning the skills of debugging are embodied [18], like pointing at the computer screen or tracing the flow of information through the wires with their finger, and in connection to their language [19].This focus on teacher moves builds on the view that teaching is inherently improvisational "because the flow of the class is unpredictable and emerges from the actions of all participants, both teachers and students" [57] (p. 2).

4 Methods

4.1 Study Overview

To study how teachers support individual and small groups of students during debugging moments, we focused our analysis on the complex interactional work they engaged in during various stages of students' debugging process. We analyzed audio and video recordings from five teachers' classrooms in two different participating districts who taught the same curricular unit and physical computing technology with their secondary students. Performing Interaction Analysis [29] we used debugging moments—or sets of interactions starting when the teacher approached the student(s) and ending when they leave the group completely whether the bug was fixed or not—as analytic segments to characterize teachers' orientations and approaches to supporting the debugging process and the recurring interactional moves that produced them. Focusing on these moments offers a new perspective on the improvisational nature of teachers' work [51] in supporting debugging as a situated inquiry [18, 59]. It also allows us to characterize the wide variety of teachers' approaches and orientations to bugs and the processes of debugging.

4.2 Study Context

This study is part of a project that, since 2017, has been collaborating with a large urban school district as part of a **Research-Practice Partnership (RPP)** [6] to develop instructional units that integrate computing in secondary school science and STEM classes [3, 63]. The initial school district involved in the RPP (School District A) is a large urban district located in the Western U.S. During the 2021–2022 school year, the RPP expanded to include an additional site in a small rural and small-town school district in an adjacent Western state (School District B).

4.3 Researcher Relationships to Participants

All six authors are researchers who worked on the SchoolWide Labs research team at two different universities. Each author has worked with teachers in School District A over the course of this study in some capacity during **professional learning (PL)** sessions. Alexandra (author 3) and Jeffrey (author 4) have supported District A teachers in their classrooms as they implemented the project's unit. While in classrooms, Alexandra and Jeffrey also collected data including audio/video recording, survey data, and observation notes. Colin (author 1), Jessie (author 2), and Mimi (author 6) have led the professional development with teachers in School District B. Jessie has supported teachers as they implemented the project's unit in their classrooms while also leading data collection.

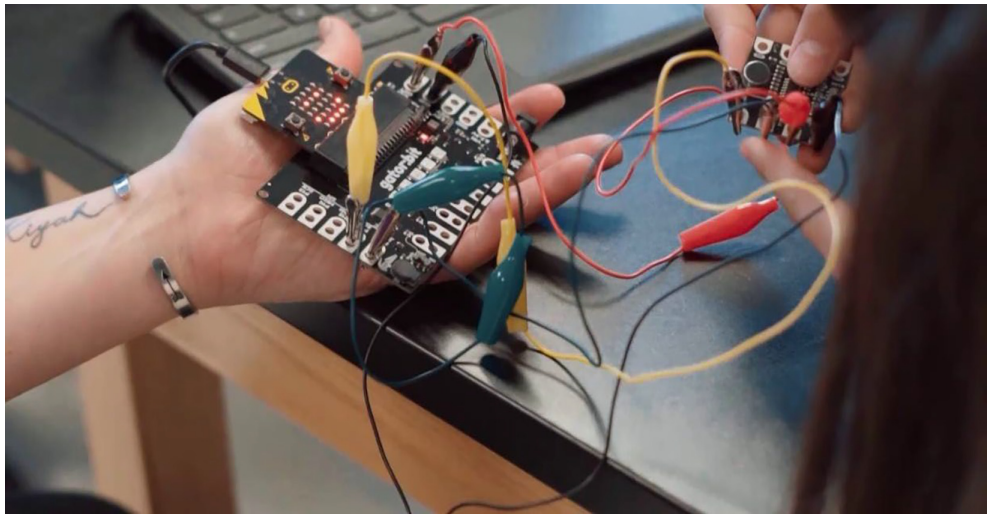

Fig. 1. A student's DASH, showing DASH components with a sound sensor connected and a smiley face produced on the micro:bit LED array.

4.4 Sensor Data Computing System

The DASH is a PCS composed of a micro:bit [68] augmented with a gator:bit. The gator:bit provides additional programmable pins on the micro:bit, which can be clipped by alligator clip connectors and a variety of alligator clippable sensors (Figure 1). Using the DASH in classrooms offers an opportunity for secondary school students to engineer their own PCS to control the entire lifecycle of environmental data, from collection, storage, manipulation, to display [20]. The DASH has up to five alligator clippable sensors, including (1) an environmental sensor that measures temperature, humidity, barometric pressure, carbon dioxide, and total volatile organic compounds; (2) a soil moisture sensor; (3) a sound sensor; (4) a UV sensor; and (5) a particle sensor (see Figure 1). The gator:bit also has a speaker and five programmable neopixel LEDs that students can use to create displays with lights and sound. The micro:bit can be programmed using the block or text-based MakeCode environment and each of the sensors has their own MakeCode blocks to control the data collection process.

4.5 Sensor Immersion Unit

As part of the RPP, researchers and educators co-designed several instructional units [3] to introduce secondary school students to the DASH [20] as a tool to support sensor-driven inquiry in science and STEM classes. Data for this article were generated while teachers implemented one of these units, the Sensor Immersion unit, which was designed to introduce students to the DASH. The five-lesson unit uses an inquiry approach (building on [55]), where each lesson sequence involves students generating questions and investigating phenomena in the world around them (e.g., sound or light levels) using data collected by the DASH. In the unit, students explore how the DASH displays data collected from sensors reading classroom sound levels, local environmental conditions, and soil moisture levels in classroom plants. As a part of the inquiry process, students generate questions about the DASH that the teacher organizes into categories corresponding to the unit's lessons. Thus, each lesson is built around answering authentic student questions about the DASH as a phenomenon. As part of answering these questions, students learn how to wire and program sensors, and how to display data that they collect. This article analyzes moments during the Sensor

Table 1. Focal Teacher's Prior Programming Experience

Name	School District	Approximate Number of Students	Grade Level	Level of Preparedness Supporting Students in Programming	Computer Science or Programming Experience
Dave	A	25	7th and 8th	Not adequately prepared	No prior experience
Gabrielle	A	20	6th	Not adequately prepared	No prior experience
Karla	B	22	9th	Somewhat prepared	Prior experience completing projects using programming and programming with students
Natalie	B	8	9th	Not adequately prepared	Prior experience completing projects using programming and programming with students
Uma	A	23	7th and 8th	Very well prepared	Computer science background and prior experience programming with students

Levels of preparedness and computer science and programming experience are based on self-reported survey data prior to each teacher's first year in the SchoolWide Labs project.

Immersion unit where teachers supported students encountering various bugs as they wired and programmed the DASH in three lessons in the unit (lessons 2, 3, and 4).

4.6 Participants

This study focuses on five secondary school science and STEM teachers (grades 6 through 9) who participated in a PL program between the 2019 and 2022 school years. We used maximum variation sampling [45] to select 5 teachers out of a total of 21 teacher participants. Our goal was to select teacher participants who had a breadth of programming experience, years of teaching, years teaching the Sensor Immersion curriculum, and debugging moments that represented an array of approaches. Although part of this study took place during the COVID-19 pandemic, we did not include any data that took place during online learning. Table 1 lists each teacher, their district, their self-reported level of preparedness for supporting students during programming, and their programming experience prior to participating in this study, as reported on a pre-survey.

Over the course of the school year, the teachers participated in several PL workshops to learn to integrate the instructional materials and the DASH into their classes [3]. Activities included collaboratively designing instructional units, modeling these activities for participating teachers, reflecting on classroom implementation, and revising curricular resources—all with the goal of supporting teacher learning, developing a teacher community, and iteratively improving instruction. In addition, four out of the five teachers took part in one PL activity focused on debugging practices where they were given buggy DASH systems, worked collaboratively to debug the system, and reflected on the process [57].

In his first year working with the project, Dave taught 7th and 8th grade science in District A. Dave worked with middle school students with a diverse set of needs and experiences. Author 4 recorded and observed (including observation notes) Dave's classroom while working closely to support him as he implemented the Sensor Immersion unit.

Gabrielle taught 6th grade science in District A. This was her second year working with the project, but her first year teaching in this school. She implemented the unit during her extra period outside of regular instruction to spend as much time on the unit as possible. All students attended this extra period at some point during the school year. Gabrielle often described her

Table 2. Video Data Sources

Teacher	Total Video for Lessons 2, 3, 4	Total Debugging Moments
Dave	180 minutes (four 45-minute class periods)	11.08 minutes
Gabrielle	180 minutes (four 45-minute periods)	8.7 minutes
Karla	150 minutes (three 50-minute class periods)	9.3 minutes
Natalie	135 minutes (three 45-minute class periods)	14.77 minutes
Uma	180 minutes (three 60-minute periods)	6.08 minutes
Total time	**825 minutes (13.75 hours)**	49.96 minutes

Bold indicates total.

own struggles and lack of experience with programming and physical computing to her students. Alexandra (author 3) worked closely with Gabrielle as she implemented the unit in her classroom and throughout her 2 years working on the project.

Karla taught 9th to 11th grade Physics in District B. Karla stated that she enjoyed integrating new technologies into her classroom and had previously introduced her students to electronic textiles. She worked with author 2 as she implemented the unit.

Natalie taught 9th grade Earth Science at STEM at an alternative high school. Her class sizes were small, ranging from six to eight students, and she had the freedom to spend more time implementing the Sensor Immersion unit. Natalie worked with author 2 as she implemented the unit.

Uma was a 7th and 8th grade STEM teacher at a school in District A. As Table 1 indicates, she reported the most confidence in her programming skills. She regularly cited her engineering degree as a source of confidence. Uma interacted with Alexandra (author 3) and Mimi (author 6) regularly during professional development sessions.

4.7 Data Sources

The primary data source for this study is approximately 14 hours of video recordings of the five classroom teachers and their students during classroom implementations of lessons 2, 3, and 4 of the Sensor Immersion unit when students engaged in programming and wiring the DASH (see Table 2).

During classroom implementations, researchers positioned one video camera at the back of the room to provide a wide view of the classroom and a second handheld camera followed the teachers around the room as they worked with students to debug their system. The handheld cameras framed the teacher, the students' faces and hands when possible, and their computer screen to record students' programs. Teachers wore a wireless lapel microphone to maximize audio capabilities in the classroom. In addition, the handheld camera meant that not all debugging moments could be fully captured by the researchers because the video was not close enough to see what was happening on the screen. Debugging moments with audio or video complications were removed from the data corpus. From the remaining handheld video recordings, we located a total of 49.96 minutes of debugging moments from all the teachers (see Table 2 and Appendix A).

4.8 Data Analysis Methods: Distilling the Framework Dimensions

To understand how teachers interacted and supported students during the debugging, we used a grounded approach to data analysis [7]. We started by co-viewing numerous instances of debugging from three different teachers (Dave, Karla, and Natalie) who participated in our program

during the 2021–2022 school year. These teachers were selected because their classroom video and audio recordings captured well-sustained interactions with groups of students as they debugged. Recordings from two more teachers (Gabrielle and Uma) who participated during the 2019–2020 school year, their first year implementing the Sensor Immersion unit, were added for maximum variation sampling.

4.8.1 Moment Selection. We consider a *debugging moment* to involve the moment the teacher approaches the student(s) until the teacher walks away. During the initial data review, we chose to classify a few instances where the teacher walks away and quickly returns to also be one single debugging moment because the student and teacher continued working on the same issue in short succession. We chose to focus on debugging moments that lasted more than 35 seconds, eliminating moments where teachers checked in with numerous students in quick succession and where little debugging occurred. The 23 selected debugging moments ranged in time from 39 seconds to over just 5 minutes and occurred at different moments throughout the Sensor Immersion unit. Appendix A lists all 23 debugging moments, and their length, date, type, and bugs.

4.8.2 Analysis of Moments: Dimension Development. From an initial review of a sample of debugging moments, we recognized that teachers displayed an array of approaches that didn't fit neatly into instructional categories as they often moved back and forth between different approaches in a given moment. From initial grounded analyses of these moments, we distilled dimensions to characterize these different approaches. This grounded approach, based on the foundations of Interaction Analysis [29], allowed us to craft dimensions based on the naturally occurring activities taking place in the classroom.

The debugging moments placed on the dimensions are not intended as a representative sample of all debugging interactions. Further, we did not conceptualize these dimensions as value judgments of teachers' debugging pedagogies. Rather, the dimensions were distilled from analyses of teachers' practice as a way to characterize the various forms of supportive interactions teachers took with students in the process of debugging the DASH. Further, the two dimensions are not the only dimensions of debugging pedagogy but ones that emerged during the debugging moments we analyzed.

To distill these dimensions, we used Interaction Analysis [29] to analyze the 23 debugging moments from 5 teachers, paying close attention to the gestures, facial expressions, and physical movements that align with each teachers' utterances and how they were subsequently taken by students in the moment. Interaction Analysis allowed us to examine the social and material ecologies surrounding the debugging moments recognizing that both knowledge and action are socially situated [23, 29]. As bugs are interactionally identified [12], we valued a socially situated analytical approach to categorizing teachers' debugging moves. Further, Interaction Analysis provides methods to examine video data in a micro-genetic way through repeated viewings—both as a research team and individually—that sketch how members of an interaction create relevancies and consequentiality. Hall and Stevens [23] describe how analysts "move through the transcripts (and video) turn by turn seeking to see what one turn set up for subsequent turns and what those subsequent turns do with prior turns" (p. 79). For our study of teachers interacting with students, this means repeatedly viewing the videos and transcripts to characterize interactional moves (i.e., teacher moves) and how they are subsequently taken up in the interaction (i.e., student uptake of teacher moves). Our analysis focuses on characterizing teachers' approaches to interaction. Such analysis is implicitly shaped by how students respond because we characterize teachers approach to the whole moment (between 40 seconds and 348 seconds) that encompasses teachers' interactional contributions, students' responses to them, teachers' subsequent responses, and so on.

Table 3. Analysis Process: From Distilling Framework Dimensions to Clustering Debugging Moments

Steps	Data Analysis	Data Sources
Step 1	Authors 1 and 2 reviewed classroom data and debugging moments	Video recordings of debugging moments that had interesting or innovative teacher moves from the Sensor Immersion unit for lessons 2, 3, and 4
Step 2	Research team (all authors) collaboratively viewed debugging moments to categorize and iterate on the dimensions	
Step 3	Larger research project (including non-authors) collaboratively viewed debugging moments to categorize and iterate on the dimensions	
Step 4	Authors 1 and 2 individually categorized 23 debugging moments on the two dimensions and compared. Moments not in alignment were reviewed together to come to a consensus.	23 debugging moments taken from video recordings of 5 classroom implementations of the Sensor Immersion unit for lessons 2, 3, and 4
Step 5	Research team reviewed 5 of the 23 moments for further categorization and consensus. Dimensions were refined and moments were clustered.	

Each video recording was transcribed using methods in line with Interaction Analysis [23, 29]. We then engaged in multiple processes of co-viewing the debugging moments as a research team seeking to understand what teacher moves were being made and how they were being taken up, individually placing these moments along our developing dimensions, iterating on their transcripts, and repeatedly discussing our interpretations as a group until we came to consistent agreements as a team. The details of our process, from defining the dimensions to selecting and clustering 23 debugging moments, are described in Table 3. We analyzed debugging moments for patterns and themes, but also for contrasts and irregularities [9]. We recognize that our analytic process with these qualitative data privileges the perspectives of researchers and not the teachers whose moves we were analyzing.

5 Findings

5.1 Teachers' Views of Debugging during PL Activities

Our prior work on debugging-focused PL activities surfaced that teachers often felt uncertain about how to best assist students in debugging PCSs [47]. Teachers expressed uncertainty about what to do if they ran into a situation where students encountered a bug they could not find or fix. For example, one teacher vocalized her uncertainty around debugging by saying: "do I have to know what I'm doing to be the teacher?" This instance reflects the tension many teachers face as they consider their role in working with students toward debugging PCSs: They often want to feel confident debugging themselves before supporting students debugging yet they rarely have the time to become experts, nor will they ever know all of the potential issues that could arise. Further, numerous contextual factors (including time in class period, number of students, and additional responsibilities) often make teachers unable to spend large amounts of time with all students to fully investigate the issue. Therefore, teachers make many in-the-moment decisions that develop dynamic approaches to supporting students' debugging in the limited time and space available.

To better characterize the varied approaches teachers took to support students, we set out to categorize teachers' moves exhibited while they supported students during debugging moments. As described next, our analysis distilled teacher debugging moves across two dimensions:

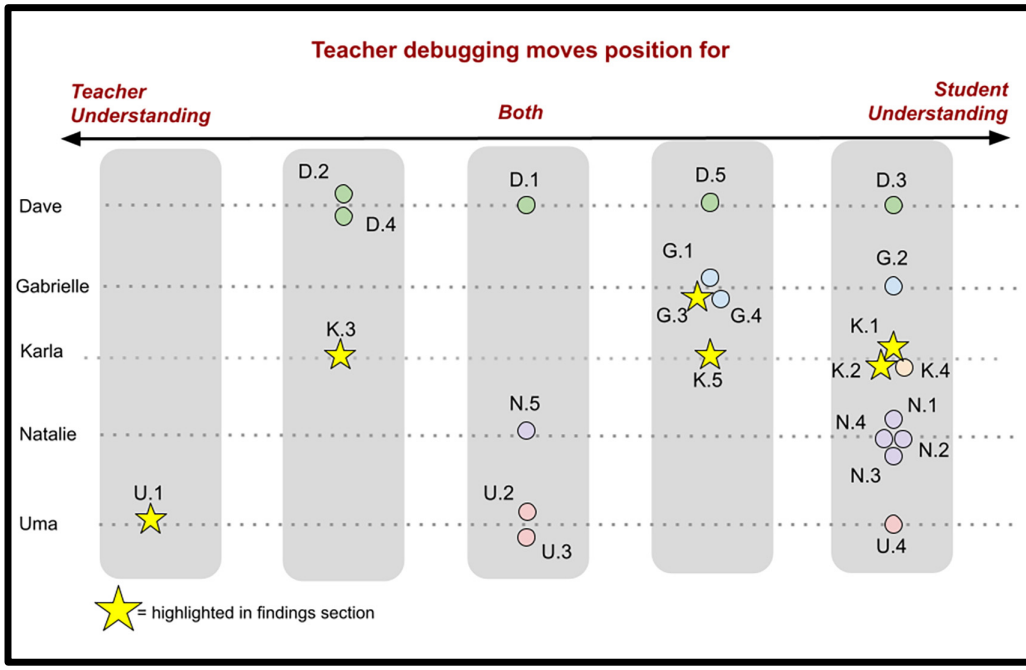

Fig. 2. Categorization of the 23 debugging moments for each teacher on the understanding dimension.

(1) a dimension that characterizes teachers' positioning during debugging and (2) the process-product dimension that characterizes if interaction was primarily organized to teach the process of debugging or prioritize the product of a debugged system. Within these nuanced dimensions, we created two clustered ordinal scales. Clustering allows the scale to be used for analyzing moments from each individual teacher.

5.2 Dimension 1: Debugging Moves Positioned between Teacher and Student Understanding

Our analyses revealed that when working with students engaged in debugging, teachers' interactions ranged from moments positioned for teacher understanding—where teachers prioritized their own understanding—to positioning for student understanding—where teachers prioritized helping the student understand the issues and underlying causes. Figure 2 shows, for each teacher, where each of their debugging moments fell on this dimension. Individual teachers used varied approaches, which often depended on the complexity of the bug and previous experience with similar bugs. Debugging moments placed on the left of the continuum in Figure 2 reflect moments where teachers worked to understand the bug for themselves. Moments positioned in the middle of the continuum reflect sets of interactions where teacher moves supported both teacher understanding and student understanding. These balanced moments often involved teachers first working to understand the bug for themselves then shifting to guide the student toward debugging for themselves. Moments at the far right of Figure 2 reflect instances where teachers exclusively centered student understanding of the bug, the process of debugging, or both. Moments starred in Figure 2 are analyzed as examples here in the Findings section (Section 5.2).

In our sampling of the 23 debugging moments from 5 teachers, a large majority were placed toward the student understanding side of the continuum. While this is not a representative sampling

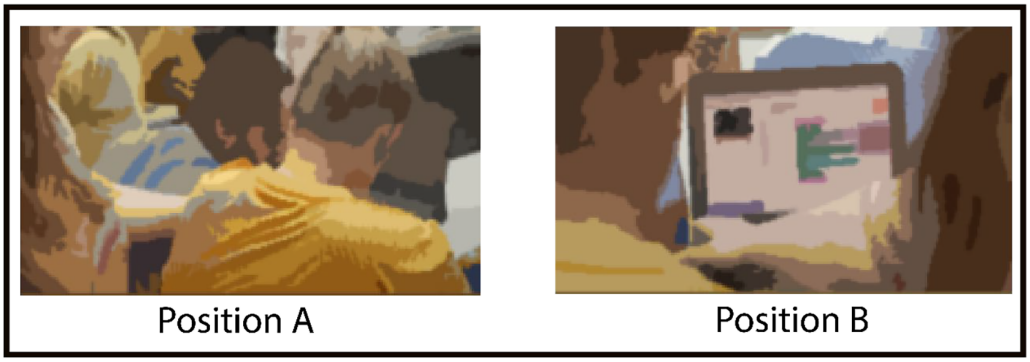

Fig. 3. Uma's movements during U.1.

from the whole data corpus, it shows that teachers in our selected moments more often prioritized moves that supported student understanding in some way.

5.2.1 Positioning for Teacher Understanding. Teacher moves that were characterized as moments positioned for teacher understanding often involved short and direct closed questions at the start of the interaction. Pryor and Crossouard [53] defined closed questions as ones "where there was a clear idea (at least for the teacher) of a correct response" (p. 4). In our analysis of teacher moves, we denoted closed questions as questions with a limited set of expected responses, usually formed as yes/no responses to questions or confirmation questions that have a clear intended set of possible responses (e.g., "what sensor are you programming for" (D.1, D.2)). Teachers regularly asked things like "did you download the code [onto the micro:bit]?" (D.2), which offered an opportunity to learn more about the DASH's current state from the student.

Teacher moves that characterized positioning for teacher understanding did interactional work to get the teacher up to speed on the student's view of the situation. These included physically positioning themselves in direct relation to the computer in order to see the program or the DASH. For example, in some approaches, the teachers either shifted the computer to read the code, took control of the mouse to search through the code, or picked up the DASH to examine the hardware wiring.

Often, teachers appeared to use these interactions to develop their own expectation of what the DASH should be doing in order to compare to its actual output, thereby interactionally producing the presence of a bug [11] in coordination with the students and the system. Throughout the 23 moments we analyzed, there were very few instances where the teacher's moves were wholly oriented for their own understanding; many were paired with interactions where the teacher was positioned for student understanding in the same moment.

In one example moment (U.1) placed far to the left at the teacher-understanding end of the dimension, a student asked for Uma's help saying "my micro:bit isn't showing up on my computer." With little verbal communication, Uma walked to the opposite side of the student (Figure 3, position A), moved an obstacle out of the way, positioned herself in front of the computer (Figure 3, position B), and took over the mousepad to find the issue. The student followed along with his gaze as Uma took an alternative path to download the program onto the computer.

The two of them did not talk about what caused the problem or why Uma had to take a different approach to downloading the program, leaving the student with only knowledge of what he saw if he ran into the same issue in the future. Here, Uma's embodied teacher moves (e.g., positioning herself and the computer for her own view, taking control of the computer) and her lack of moves

Fig. 4. Natalie's movements during N.5.

like verbalizing what she was doing meant we characterized this moment as positioning for teacher understanding. Her choice to focus on developing her understanding of the bug could have been due to the type of bug, a failure of the micro:bit's communication with the computer.

5.2.2 Positioning for both Student and Teacher Understating. We placed moments toward the middle of the dimension when teachers seemed positioned both for their own understanding and their students' understanding. These moments arose from two distinct approaches. Teachers either (1) worked through co-discovery of bugs with each interactional move positioned simultaneously for teacher and student understanding or (2) moved back and forth between positioning for teacher and student understanding. In our analysis, teachers who deployed the latter approach often started positioning inquiry for teacher understanding and then shifted to position inquiry for student understanding, possibly once they had a vision of what the bug might be.

The following example from Natalie (N.5) exemplifies the co-discovery approach with one student working on one DASH. Natalie (N.5) spent 5 minutes and 18 seconds with one student. The student tried to program the sensor to display both humidity and pressure but got stuck trying to use the logic blocks to display two variables. Natalie told the student that she wasn't sure how to do this as she had never done this herself. Sitting side by side with her student, Natalie positioned the computer equally so they each could look at the program together, taking turns driving the mouse, pointing to lines in the code, and sifting through the available blocks they could add to the program (see Figure 4).

Natalie also modeled her own thinking, questioning, and debugging practices such as reading the code aloud, having the student pull up the tutorial as a guide, moving back and forth between what the tutorial explained and the student's code. Natalie's language also displayed a sense of co-discovery as she regularly used "we" to describe the actions and movements in the interaction (e.g., "I think we want to put something in there" (pointing to code) and "pretty sure we make a dummy variable named switch which shows if we are showing humidity or pressure").

In sum, Natalie's approach worked simultaneously to clearly help her understand the bug and to support the student to debug similar issues in the future by having strategies and skills to deploy.

Therefore, we characterized this moment as in the middle of positioning for student understanding and teacher understanding.

5.2.3 Positioning for Student Understanding. Moments characterized as positioning inquiry for student understanding were marked by three types of teacher moves: (1) scaffolded questions and directives that encouraged students to take an active role in locating and fixing bugs, (2) verbal guidance that supported students in noticing issues and possible fixes themselves, and (3) clear embodied gestures connected to the computer and the DASH, allowable because of the teachers' proximity to both the DASH and the computer without getting in the view of the students.

Teacher questions and directives encouraged students to take an active role in locating bugs, talking through their thinking, reading the code aloud, and making sense of their expectations of what the code should be doing. For example, Karla repeatedly asked students: "what are you telling it [the micro:bit] to do?" (K.1, K.2, K.3) and "what instructions did you give it?" (K.2, K.5). Through answering her questions, Karla could uncover the places where students were confused, find gaps in their understanding, and scaffold approaches to help students understand the bug. Simultaneously, through talking through their code students had to make sense of what the code was supposed to be doing, learn more about programming, and develop a more robust expectation of what the DASH should do.

In these moments, teachers often worked as guides, helping students through asking open-ended or exploratory questions or proposing directives to promote student understanding. Sometimes this occurred after the teacher first determined what the bug was for themselves and then shifted into student-centered understanding. Other teachers entered each encounter with students from a student-oriented approach sometimes having no idea of what the bug was. As one teacher, Natalie, explained in a post-implementation interview: "I have no idea what I'm doing, so I just ask a lot of questions."

Being a guide also meant pointing students to additional supports such as tutorials or example code so students might learn about the variety of resources they could use when stuck in the future. In one moment (G.3, Figure 5), when Gabrielle went to help a student, she recognized a potential bug right away; she looked at the screen and saw that there was an empty space in the "or else" block (line 6). Gabrielle's quick identification of a potential problem helped her target the learning moment toward student understanding. Gabrielle encouraged her student to pull up example code (line 9) and compare his code with the example code to locate the issue (line 13). Gabrielle offered the example code as a visual tool while also reading the code aloud (lines 14–15, 20–21), asking questions (lines 11, 18–19), and directly pointing to lines in the code to direct her student to the issue (line 6). We thus categorized Gabrielle's interactions on the far right of the dimension's continuum because the entire interaction was meant to help the student notice the bug rather than pointing it out to him.

Noticeably, teachers like Gabrielle also embodied student-understanding centered approaches differently, using their bodies and gestures to orient students toward debugging. Gabrielle, for example, pointed to each line in the code as she read aloud both the example code and the student's code as the student followed along. She also simultaneously pointed at an issue in the code and said "Uh oh" to direct the student to notice the error (line 20), a regular move we identified in other work [24].

Karla's actions in K.5 demonstrated a clear shift in body posture and orientation when she moved from positioning for teacher understanding toward student understanding. She originally leaned toward the computer and pulled the laptop toward her so she could see if the students had downloaded the correct program (positioning inquiry for teacher understanding). She then quickly shifted her approach, moving the laptop back toward the students, and directing students toward

1	**Gabrielle:**	K what's up.
2	**Nate(s):**	((points to the screen)) Um ok. I already coded this one ((grabs the micro:bit still plugged into the computer)).
3		[I just... ((inaudible))
4	**Gabrielle:**	[OK did] you ((touches Nate's shoulder with index finger)) download your code.
5	**Nate(s):**	Yeah. I downloaded it=
6	**Gabrielle:**	=OK now you got a big fat space there. You see that space there ((points to the computer screen)). When
7		you're writing code, you can't leave stuff like that. ((Nate looks up to Gabrielle and then to camera)).
8	**Gabrielle:**	K so let's ((moves to the other side of Nate and turns the computer to orient viewing for both of them)). Did
9		you look at the um at the sample code?
10	**Nate(s):**	Ok. ((pulls up sample code on Makecode tutorial)) Like that.
11	**Gabrielle:**	Mmmk. So this says -- ut oh this is number 6. Did you already download up to number 5?
12	**Nate(s):**	Yeah.
13	**Gabrielle:**	O:::K so you're gonna go for it. Huh. I like it. OK. So then this is what the code is supposed to look like. K so
14		what we've got it says on button B, so when button B is pressed, if the sound intensity is less than four
15		hundred -- that's good, you changed that huh.
16	**Nate(s):**	MMhmm.
17	**Gabrielle:**	K. Um then you're gonna start melody repeating once. K and then it say:::s. What does that say, shoestring
18		music. OK. Let's go look at your code. ((Nate switches window back to just his code)). MMk. Ut what does it
19		say? Can we move it down so we can look at the whole thing?
20		((Nate scrolls up a bit)). Ut oh ((Nate switches the button press block)). There you go you found one thing. K
21		then show number sound intensity. If the sound intensity. K. Start melody ((Nate clicks on the block to change
22		the melody)). It could -- it's OK. It doesn't matter which one you use. You could use any one. ((Nate clicks off
23		the box)). K repeating once. or else WHAT.
24		(5s) Go back and look at your sample code. ((Nate clicks hint button in tutorial)). K uh what do we got here.
25		Oh, we don't need an "or else." We don't need an "or else." So go and change that one. We don't need an ["or
26		else."
27	**Nate(s):**	((points to logic block on the screen)) [would]
28		it be that one.
29	**Gabrielle:**	There you go we don't need an "or else." ((Nate begins changing code to put new IF statement in)). Look how
30		smart you are.
31	**Nate(s):**	I take this ((pointer is on the play melody block)) right.
32	**Gabrielle:**	That's OK. Yup take those out. Great. Keep going.

Fig. 5. A snippet of the transcript for G.3.

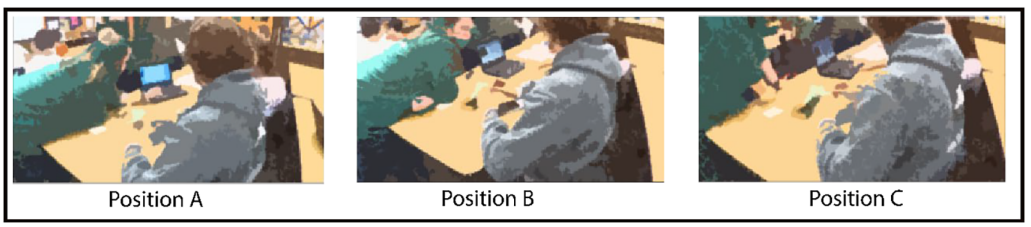

Fig. 6. Karla's positioning for K.5.

the next steps saying "okay, will you click this one and drag it over here to where it says micro:bit." After the download was complete Karla realized the issue and her body position, questioning, and approach shifted. She started asking questions like "okay, it's waiting for you. What is it waiting for you to do?" and "what instructions did you give it?" While previously Karla's gaze was focused on the screen itself so she could see the program (Figure 6, position A), here her eye focus was on the students rather than the computer (Figure 6, position B). Karla's physical posture changed also as she stood up from her position where she could no longer see the screen but rather focused on the students again asking "so what are you thinking? What should we do?" (Figure 6, position C). If everything after the shift in Karla's approach were its own debugging moment, we would have characterized it as purely positioning for student understanding and placed it on the far right cluster of Figure 2. Instead, we positioned it one cluster to the left of the full student understanding cluster.

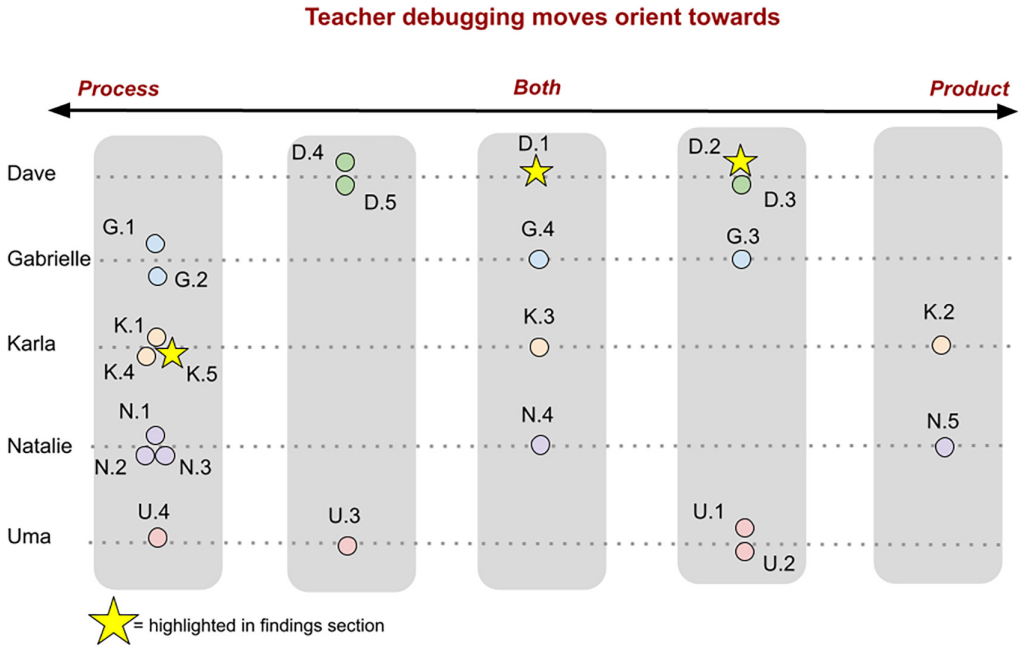

Fig. 7. Categorization of the 23 debugging moments for each teacher on the process/product dimension.

5.3 Dimension 2: Inquiry Orientations between Process and Product of Debugging

Our analysis uncovered how, in supporting students who were stuck, teachers oriented their inquiry toward finding and fixing bugs in different ways as they interpreted students' developing relationships to the DASH. Starting from the data, we distilled a dimension ranging from the teachers' inquiry as fully process oriented—supporting students to see and develop a process for finding and fixing bugs—to fully product oriented—supporting students to make the DASH work as intended (see Figure 7). Much like the characterization of the different kinds of debugging learners by Perkins et al. [50], this dimension captures the different approaches teachers take to support debuggers, which may (to some degree) rely on their implicit read of the student's current role as Tinkerer, Mover, or Stopper. Each extreme of the dimension indicates if almost all (if not all) of the inquiry was oriented toward either the product or the process of debugging. Points in the middle indicate that either there was a shift in the kinds of inquiry interactions (questions and directives) teachers initiated across the moment or the teacher's inquiry interactions were, themselves, characterized as both process and product oriented.

5.3.1 Process Oriented. Teacher debugging moves that helped to characterize a moment as process oriented typically positioned the teacher as an investigator with the student through open-ended questions, explicit inquiry strategies, and/or directives. For example, some teachers narrated what they were looking at and what they were looking for, as they engaged with the DASH that the student identified as buggy. Other examples included teachers asking open-ended questions that could lead students to engaging directly with the DASH without the teacher. Alternatively, teachers sometimes directed students to perform an open-ended task (e.g., "program it to show the temperature"). These moves develop opportunities for students to notice and develop a process for debugging or the underlying explanation of a bug, rather than just noticing the bug and/or the steps to rectify it.

One example (K.5) depicts the kinds of moves that helped characterize a moment as process oriented. This moment was also analyzed in Section 5.2.3. In this moment, coded solely as process oriented, Karla spent a total of 3 minutes 45 seconds with two students. She provided plenty of time for the two students to talk through their thinking and look through the program, focusing on the importance of debugging time as useful, productive problem solving by repeatedly asking open-ended questions. On multiple occasions Karla asked students "What instructions did you give it?" (K.5) which prompted them to re-examine the code and explain what they thought was going on. She continued to ask similar questions throughout her engagement with these two students especially when there were disagreements between the students or when students' answers or questions demonstrated a lack of understanding or awareness of what to do next. Karla focused on the process even after the students figured out the solution to the bug. One student read the code adding "oh, so we shake it, the thing moves?" (K.5). Karla responded by helping them think through the next step "so how do we send that information to here?" and "then what?" (K.5) thereby pushing them to notice and understand the process involved in ensuring the correct code was displayed on the micro:bit. In other words, even though the bug was already fixed to a degree, Karla further pressing the issue highlights the importance of downloading the program onto the physical system, possibly pushing for it to become regular check in their future debugging process. Karla's move to extend the facilitation to account for possible other bugs underscored that the moment should be far to the left of the dimension's continuum on Figure 7. Overall, Karla's discourse moves (asking open-ended questions, reviewing what to do even after they solved the bug) demonstrated a clear focus on the process of debugging with these two students.

5.3.2 What Does It Mean to Be Process and Product Oriented? In a few instances, we characterized teachers' orientation as balanced between product and process when they helped students debug the DASH. In each of these moments, the teacher produced interactional moves that oriented the student to some sort of process for finding the bug but, eventually, focused on just getting the DASH working.

For instance, we coded multiple moments from Dave's class (D.1 and D.2) where his students ran into the same DASH symptoms one period apart: the micro:bit displayed a zero when it was not supposed to display a number at all but display a symbol according to the conditional logic they programmed. We coded D.1, the first time Dave encountered this specific bug, as directly in the middle between process and product oriented. In this moment, Dave (1) tested the system with the student, (2) offered possible sources of the issue (e.g., "it's either a wiring issue… it could also be a power issue"), (3) checked the wiring with the wiring diagram by narrating what he was looking at, and finally (4) offered a possible solution. Dave described this possible solution as "Let's dump an old code in just to—like—delete what you're trying to write and see if we can use it [after]." Although this offered solution did not help students understand the root cause of the issue, it did fix the DASH which led to Dave to use the solution each time it showed up again (e.g., D.2). Similar to the advice often given when someone's computer isn't working correctly—turn it off and turn it back on—this directive helped students get unstuck and continue investigations, but it did not offer much opportunity for them to learn how to debug the system. This finding aligns with Fitzgerald et al. [17], who describe a common strategy for novice debuggers to fix bugs by removing pieces of code instead of investigating the code. It is also an example of a teacher's strategy when they are unsure of what to do next to help students find success in getting the system to work.

We categorized this moment (D.1) in the middle of the dimension because Dave shifted between an earlier more process orientation of inquiry (testing the system and then posing "its either a wiring issue… it could also be a power issue") toward a product orientation (offering a fix by "dumping old code…"). We identified this shift when his actions moved from seeking to help the

student notice a possible issue in the wiring or power to suggesting they reset the DASH and test to see if this fixed the problem. At first, Dave was clearly oriented toward helping the student develop a process of noticing bugs and fixing them by describing what the issue could be (a future strategy) and then using the wiring diagram in a demonstrable way to support the student using it in the future (he narrated the color of the wires he was looking at on the diagram and then on the hardware). However, he then stepped away to support another student for about 40 seconds. When he came back, he said "so this is a power issue," before directing the student to disconnect power and reconnect to the laptop for power before testing again. By being more direct but still offering reasoning, he started to shift his orientation toward just getting the DASH to work. When this did not work, he offered a possible fix without helping the student notice what the bug might be.

Other teachers had similar shifts when time in class was running short (as it was here) or, maybe, they wanted students to feel successful in getting the DASH to work. Dave's moments with this particular bug really pushed our analysis to characterize a balance between both orientations because the initiative to develop a process was there but the situation did not allow for it.

5.3.3 Product Oriented. During some moments when students seemed to be stuck, teacher moves oriented the inquiry more toward getting the problem fixed than helping the student develop a debugging process. Moves that exemplified this orientation included asking closed questions specifically about the DASH setup, giving very specific directions about what to do next, and taking actions like that described in the previous section—resetting the DASH that might fix the issue before knowing the source of the bug.

When Dave encountered the same issue his students had run into in the previous class period (D.1) a second time, he pursued a product-oriented inquiry. In the moment we call D.2 (transcribed in Figure 8), Dave's questions and directions were closed, meaning they limited the range of possible responses (e.g., a yes/no answer or a direction to do a very specific task). For example, the question Dave posed ("we feel good about programming," line 2) is clearly a yes or no question positioned to give Dave an opportunity to see whether the group had considered an issue in the code that could be explored further. This led to Dave giving the students a directive on how to solve the issue (line 13) which he experienced in the previous class. His directive in line 13, "dump an old code on. See if we get rid of that zero. And then we can try your code again," accompanied by "see if it gets rid of that zero" (line 26) confirms the clear focus on fixing the issue rather than demonstrating or engaging in a replicable inquiry to find a bug and fix it. Essentially by trying to reset the DASH, Dave's directive pointed the students toward finding an old program that worked, running it on the system, and then re-running the code that was not currently working. The specific question about the DASH and clear directives position Dave's inquiry as product oriented, attempting to first figure out if this was the same issue as before and offer a fix, without even knowing the source of the issue.

5.4 Contextual Factors Influencing Debugging Moves

As we reviewed the video data, it became increasingly clear that the moves teachers took were contingent on several contextual factors including (1) time left in the class period, (2) the specific bug, (3) the teacher's ability to initially identify the bug, and (4) the number of students looking for help in the class. For example, toward the end of the class period, one teacher may decide to shift from orienting to the debugging process to just finding and fixing the bug so the student feels successful. Even teachers' access to materials and the number of classes taught had a clear impact on their pedagogical choices and in-the-moment moves: For example, needing to have the same 20 DASH setups for the next class period often meant that students could not come back to a buggy setup the next day.

```
Sam(s):   I don't know what we did.
Dave:     Alright. We feel good about programming.
Sam(s):   Uh [tha
Dave:        [That zero] tells us we've got some kind of error. Let's
          go take a look. See if we can figure it out. ((they both
          walk back to Sam's seat)).
Dave:     So. We feel good about the programming. And we feel good
          about the wiring.
Sam(s):   Yeah
Dave:     But we're getting a zero.
Simon(s): Wait are you supposed to do an all of -- like aren't you
          supposed to do like all like three bits of like program.
Dave:     Uh huh. What I-- what I suggested we tried with my last
          class. Take an old code. Dump an old code on, a really basic
          code. See if we get rid of that zero. And then we'll try
          your code again. So um=
Sam(s):              =I know we tried doing that and then we:: do
          that or like=
Dave:                =you can go back to:: the homepage for the
          micro:bit. And it's stored all of your old codes that you've
          written. Pick one.
Sam(s):   Pick no.
Sam(s):   Um. This one.
Dave:     Did that one work.
Sam(s):   Yeah.
Dave:     So, connect download. See if it gets rid of that zero. That
          was -- that was how my group troubleshooted last period and
          and it ended up working.
```

Fig. 8. A snippet of the transcript of Dave (D.2) working with two students, denoted by (s).

The number of students in a class was an important contextual factor influencing the debugging moments (see Table 1 for additional contextual factors). The class size differences between Dave's and Natalie's classroom contexts exemplify how teachers must make different choices that affect how the debugging moments unfold along the dimensions we identify above. Dave taught a class of 25 middle school students who were constantly looking for support from him. None of his debugging moments with students that we observed lasted more than 130 seconds without being interrupted (both D.1 and D.2 were two-part moments because he quickly came back to the same students). Natalie, however, taught a smaller class (ranging from six to eight students daily) with more time dedicated to the unit. Therefore, she was able to have more time to devote to debugging moments, affecting how she decided to interact with students and the DASHs.

We aim to not place value judgment on any of the situations our teachers were in, nor the choices teachers made in these moments. Instead, we highlight that the two debugging pedagogy dimensions capture the breadth of possible choices and experiences teachers can have in supporting students to debug, while also supporting them to learn debugging processes and skills.

6 Discussion

This study examined teachers' interactions as they orchestrated secondary school classrooms during small group work time when students encountered bugs as part of an introductory physical computing unit in a range of grades (6–9). To understand teachers' approaches to supporting their secondary school students, we asked: *What moves do teachers make to help support students in developing debugging practices?* We then distilled two dimensions of debugging pedagogies that

characterize teachers' interactions with students: their positioning for understanding and their orientation for inquiry.

As computer science education is increasingly integrated into K-12 classrooms and often facilitated by teachers with limited content knowledge and computing background, our two dimensions—developed from analyses of the practices of secondary teachers—serve as a foundation of a framework to help novice teachers learn to support students to successfully engage with PCSs. As the work of Dave and Karla—and our other partner teachers—exemplifies, teachers do not necessarily have to be programming experts to support students in debugging their PCS. Debugging skills and programming knowledge, while important in helping students learn to debug to some degree [30], are alone insufficient resources [41]. Teachers encounter a variety of demands in the classroom that they must attend to during debugging moments [24], which require knowledge of and experience with debugging pedagogical moves. As our analysis indicates, these moves include a variety of verbal and embodied inputs to support students in the two-pronged goal of (1) getting their systems working and (2) developing skills as debuggers themselves.

Secondary teachers who are new to computing and debugging need resources that attend to the demands, contextual factors, and complex environment of the secondary classroom. However, much of the extant research on debugging has either focused primarily on students' debugging practices (e.g., [13, 44]) or debugging in college-level CS courses. Debugging support for classrooms must be designed for teachers who may have little programming experience while also being sensitive to the contextual factors that impact teachers' pedagogical decisions in the classroom. Because bugs are interactionally produced [16], debugging moments are also highly unique and contextual. The teachers in this study—independent of their programming expertise—entered moments of uncertainty when students were stuck by using strategies such as asking open-ended questions, pointing students to resources, or working with students as a guide or through co-discovery. In this way, the teachers used the experiential resources they had—both about debugging and about supporting student inquiry—to support student learning to debug. PL for integrating computer science into secondary classrooms must engage teachers in calling on and developing their interactional toolkits for making appropriate decisions in-the-moment with students and the material tools at hand. We articulate the implications of this in the next section.

7 Conclusion and Implications

This study offers several contributions to the field related to how teacher uncertainty, differences, and context influenced resulting interactions. The teachers' practices depicted in this article hold implications across the STEM disciplinary spectrum informing innovative practices and pedagogies supportive of student inquiry [41] and epistemic agency [46] as they deal with failures and roadblocks. In particular, the framework described in this article further defines what debugging pedagogy, from a sociocultural perspective, looks like in practice. It also illustrates how teachers negotiated moments of uncertainty and provides insights into embodied interactional moves that assist teachers in supporting students' developing their own way forward when something is not working. Finally, the work provides a grounding for further developing teachers' and researchers' studies of debugging pedagogies. Each contribution is discussed next.

7.1 Sociocultural Debugging Pedagogies

Longer periods of open-ended inquiry for students to assemble, wire, and program their DASHs offered more opportunities for inconsistencies to arise between student expectations and DASH operation, requiring teachers to work with students at different stages in the debugging process in close proximity. This kind of teaching is inherently improvisational [47]. Our five focal teachers had to make decisions about how to proceed in the moment to support students through the evolution

of their debugging skills, much in the same way that the authors of [46] urge science teachers to support students in the inquiry processes inherent in being "doers of science" rather than "receivers of facts."

We started from the assumption that debugging is a situated inquiry [59], where the debugger recognizes and fixes an inconsistency between their expectation of the system and the system itself. The debugging moments from these five teachers help to articulate how learning this kind of situated inquiry requires, from a sociocultural perspective, the teacher role to be emergent, relational, and explicitly dependent on where students are in relation to the materials they are working with [41]. In analyzing these moments, we had to focus not only on the teachers' moves (i.e., what the teacher said and did) but also on how they were taken up by the students, where the inquiry of the group went, and subsequent teachers' moves were produced.

Our analysis provides evocative examples of the different kinds of choices teachers make in these moments that are informed by their expectation of how students can learn to be debuggers themselves. Implicit in this are two sometimes competing goals of getting the PCS working and supporting students in developing a "debugging philosophy" [46] where they are open to learning from being stuck. Debugging pedagogies, or the perspectives on supporting students learning to become debuggers, relies on these two tensions in the interactional moments of practice. For example, the whole of Dave's interactions in D.2 (characterized as more product oriented) displays a pedagogical choice to support the students by getting the system working, which implies that, for him at that time, the error being presented was not worth students' time to invest in figuring out how to fix. Similarly, Natalie's positioning of the interaction as co-discovery (N.5) displays a pedagogical choice to take the time to work with students while implying that learning to find and fix this kind of bug is important. As we articulate above, we assume that pedagogical choices were informed not only by a debugging pedagogy but also by the contextual factors at play in a particular interactional space and time.

Thus, the nuances of debugging pedagogies cannot be completely represented in the orientation or process/product dimensions. Yet, these nuances are necessary to understand what debugging pedagogies are essential for students learning to program and work with PCSs like the DASH. Further, being good pedagogues is not inherent in being expert programmers. Teachers need a certain level of understanding of the system and a set of pedagogical skills that adapt to where students are, offering scaffolds and challenges that orient students toward debugging the current system for themselves while developing skills to debug in the future.

7.2 Approaching Uncertainty

The co-construction of the debugging moments between teachers and students required both deep interaction and responsive creativity on the part of the teacher [10, 51]. From a sociocultural perspective, the emergence of joint meaning-making cannot be reduced to any one teacher's or student's intentions in an individual turn because the teacher "cannot know the meaning of her own turn until the other actors have responded" [58] (p. 2). Therefore, teacher–student interactions involved some level of uncertainty for the teacher.

Generally, learning tasks that are unfamiliar [13, 54], novel [25], or complex elicit additional possibilities of uncertainty for the teacher. In this study, many of the issues that arose were novel for the teachers since it was their first time implementing the unit (except for Gabrielle who had implemented a couple of months earlier with another class) or including programming into their classroom. In addition, the teachers' initial assumptions about sources of the system's failure did not always align with what others might identify as the actual problem (see D.1) because of the complexity of working with a PCS. Teachers' approaches to handling this complexity are reflected in how the moment was positioned within both dimensions.

One consequence of the uncertainty present in each moment was that there was a significant amount of variation exhibited both across one teacher's moments and all the teachers in this study. We found that no teacher's moments were represented solely on one side of the continuum for either dimension. Even within a moment, some teachers moved across the orientation continuum by debugging with a more product-oriented approach as more information became available about the cause of the inconsistency or as class time ran short.

During moments of uncertainty when helping students debug, teachers must decide: Do they simply help students to have a working system free of inconsistencies (such as moments D.1 or D.2) or do they take a more process-oriented approach by guiding the students through a more open-ended inquiry process about the system as a whole [60] as illustrated in moment K.5? In this study, teachers took a variety of approaches from offering specific instructions to change the system with the goal of resolving the inconsistency (product orientation) or guiding students through a systematic process (or processes) to help find the source of the inconsistency (process orientation). The nuances and patterns in these debugging moments were highly influenced by the context. Some examples included the amount of class time remaining (i.e., Dave), the number of students in the class (i.e., difference between Dave and Natalie), the number of students currently claiming to be "stuck" and asking for help, the kind of inconsistency that the DASH was exhibiting, and the familiarity of the teacher with the bug.

7.3 The Embodied Nature of Debugging Pedagogies in Action

To make sense of inconsistencies, teachers relied not only on words but also embodied moves during the debugging process using coordination of their bodies, hands, and gestures to orient students toward debugging processes [19]. For example, both Gabrielle and Karla supported student noticing by gesturing toward the screen to connect what they were saying to specific parts of the program on the screen. In U.1, Uma positioned herself in front of the computer while the student was to the side, illustrating a moment that exhibited an extreme end of positioning for teacher understanding. Similarly, Karla used her body position to illustrate a shift from teacher understanding toward student understanding during some of her debugging moments as she moved to decenter herself from the computer and place the focus on the students. This implies that a focus on talk alone may provide a less complete view of teachers' debugging pedagogies. Future research, therefore, should focus on the embodied nature of teachers' practices and their relationship to the debugging pathways they negotiate with students.

7.4 Toward a PL Framework for Debugging Pedagogies

While students need scaffolds to support the development of their physical computing skills, teachers also need scaffolds to facilitate productive teacher moves during the debugging process. The two dimensions presented in this article provide a basis on which to design PL experiences for teachers to help them learn how to navigate moments of struggle when students' PCS is inconsistent with their expectations. In other words, the dimensions inform designs of PL opportunities that develop secondary teachers' debugging pedagogies.

One strategy for developing debugging pedagogies involves engaging both practicing and preservice teachers in video analysis of debugging moments either of their own classroom (for experienced teachers) or example classrooms (for preservice teachers). In these video analysis sessions, teachers analyze the embodied, verbal, and gestural teacher moves and place moments on the understanding and process/product continuums. This strategy could (1) illustrate for teachers how teachers can navigate the uncertainty in the moment, (2) illuminate a variety of strategies and debugging pathways [10], and (3) support teachers in reflecting on the numerous contextual factors influencing debugging moves in each video. Such would be particularly beneficial approach for preservice

teachers to reflect on their comfort level supporting students' debugging, imagine how they might support students in their future teaching, and devise a range of strategies they can emulate as they refine their own debugging pedagogy. Additionally, preservice teachers may collaboratively view and place debugging moments on the dimensions with a mentor or cooperating teacher as an artifact to think through pedagogical choices and implications. Acknowledging that our analysis came solely from the perspectives of the researchers, this strategy could also be the next step in research debugging pedagogies through "bottom-up knowledge production" like the culturally situated design tool creation process in [14] or working with cultural experts to develop situated, culturally responsive, debugging practices for youth [38].

Another PL model to involve both novice and experienced teachers simultaneously in developing their debugging pedagogy involves a debugging activity where one teacher acts as the teacher and two other teachers act as students as they debug a PCS [47]. Whereas we previously employed this strategy concluding with a brief reflective discussion [47], the dimensions themselves could be employed as a reflective guide for teachers to analyze where their choices fell and consider why they made those choices in the moment. In general, these kinds of activities could support teachers to be more cognizant of their own actions and enhance teachers' understanding of how they can support their students to become effective programmers of PCSs. They offer teachers opportunities to reflect on the decisions they may make in-the-moment to support students and the consequences of those decisions. We see this activity as salient for PL with teachers of various levels of experience in programming as novice and more veteran secondary teachers may learn from one another's moves in the moment. Such discussion around the dimensions can help teachers prepare for future decisions supporting their students' development as debuggers, thus developing teachers' debugging pedagogies.

Acknowledgments

The authors are deeply grateful to the teachers in this study (Gabrielle, Uma, Dave, Karla, and Natalie) for giving them the gift of video data from their classroom and allowing them to continue to think with their stories (interactional and longitudinal). They also thank the people who made the two Research-Practice Partnerships possible that these teachers participated in. Further, they thank the rest of the SchoolWide Labs team—including Mimi Recker, Jennifer Jacobs, Tamara Sumner, Bill Penuel, Quinten Biddy, Srinjita Bhaduri, and others—for supporting the intellectual work, data generation, and teacher professional development that made this article possible.

References

[1] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An Analysis of Patterns of Debugging among Novice Computer Science Students. *ACM SIGCSE Bulletin* 37, 3 (Sept. 2005), 84–88. DOI: https://doi.org/10.1145/1151954.1067472

[2] Jessica L. Alzen, Kelsey Edwards, William R. Penuel, Brian J. Reiser, Cynthia Passmore, Chris D. Griesemer, Aliza Zivic, and Christina Murzynski. 2020. Exploring the Connections between Epistemic Agency and a Commitment to the Collective Enterprise of Sensemaking in the Science Classroom. In *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS) 2020.* M. Gresalfi and I. S. Horn (Eds.), 1879–1886.

[3] Quentin Biddy, Alexandra Gendreau Chakarov, Jeffrey Bush, Colin Hennessy Elliott, Jennifer Jacobs, Mimi Recker, Tamara Sumner, and William Penuel. 2021. A Professional Development Model to Integrate Computational Thinking into Middle School Science through Codesigned Storylines. *Contemporary Issues in Technology and Teacher Education* 21, 1 (2021), 53–96.

[4] Paulo Blikstein. 2012. Bifocal modeling: A Study on the Learning Outcomes of Comparing Physical and Computational Models Linked in Real Time. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI '12)* ACM Press, Santa Monica, CA, USA, 257. DOI: https://doi.org/10.1145/2388676.2388729

[5] Paulo Blikstein. 2013. Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future. In *Proceedings of the 12th International Conference on Interaction Design and Children.* ACM, New York, NY, 173–182. DOI: https://doi.org/10.1145/2485760.2485786

[6] Cynthia E. Coburn, William R. Penuel, and K. E. Geil. 2013. Research-Practice Partnerships: A Strategy for Leveraging Research for Educational Improvement in School Districts. William T Grant Foundation. Retrieved from https://rpp.wtgrantfoundation.org/wp-content/uploads/2019/10/Research-Practice-Partnerships-at-the-District-Level.pdf

[7] Juliet M. Corbin and Anselm L. Strauss. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (2nd. ed.). SAGE, Los Angeles.

[8] Maggie Dahn and David DeLiema. 2020. Dynamics of Emotion, Problem Solving, and Identity: Portraits of Three Girl Coders. *Computer Science Education* 30, 3 (July 2020), 362–389. DOI: https://doi.org/10.1080/08993408.2020.1805286

[9] Sara Delamont. 2016. *Fieldwork in Educational Settings* (0 ed.). Routledge. DOI: https://doi.org/10.4324/9781315758831

[10] David DeLiema, Jeffrey K. Bye, and Vijay Marupudi. 2024. Debugging Pathways: Open-Ended Discrepancy Noticing, Causal Reasoning, and Intervening. *ACM Transactions on Computing Education* 24, 2 (June 2024), 1–34. DOI: https://doi.org/10.1145/3650115

[11] David DeLiema, Maggie Dahn, Virginia J. Flood, Ana Asuncion, Dor Abrahamson, Noel Enyedy, and Francis Steen. 2020. Debugging as a Context for Fostering Reflection on Critical Thinking and Emotion. In *Deeper Learning, Dialogic Learning, and Critical Thinking* (1st. ed.). Emmanuel Manalo (Ed.), Routledge, 209–228. DOI: https://doi.org/10.4324/9780429323058-13

[12] David DeLiema, Yejin Angela Kwon, Andrea Chisholm, Immanuel Williams, Maggie Dahn, Virginia J. Flood, Dor Abrahamson, and Francis F. Steen. 2023. A Multi-Dimensional Framework for Documenting Students' Heterogeneous Experiences with Programming Bugs. *Cognition and Instruction* 41, 2 (Apr. 2023), 158–200. DOI: https://doi.org/10.1080/07370008.2022.2118279

[13] Kayla DesPortes and Betsy DiSalvo. 2019. Trials and Tribulations of Novices Working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research.* ACM, Toronto, ON, Canada, 219–227. DOI: https://doi.org/10.1145/3291279.3339427

[14] Ron Eglash, Michael Lachney, William Babbitt, Audrey Bennett, Martin Reinhardt, and James Davis. 2020. Decolonizing Education with Anishinaabe Arcs: Generative STEM as a Path to Indigenous Futurity. *Educational Technology Research and Development* 68, 3 (June 2020), 1569–1593. DOI: https://doi.org/10.1007/s11423-019-09728-6

[15] Deborah A. Fields, Kristin A. Searle, and Yasmin B. Kafai. 2016. Deconstruction Kits for Learning: Students' Collaborative Debugging of Electronic Textile Designs. In *Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education.* ACM, Stanford, CA, USA, 82–85. DOI: https://doi.org/10.1145/3003397.3003410

[16] Deborah Fields, Gayithri Jayathirtha, and Yasmin Kafai. 2019. Bugs as a Nexus for Emergent Peer Collaborations: Contextual and Classroom Supports for Solving Problems in Electronic Textiles. In *A Wide Lens: Combining Embodied, Enactive, Extended, and Embedded Learning in Collaborative Settings. Proceedings of the 13th International Conference on Computer Supported Collaborative Learning (CSCL '19)*, Vol. 1. K. Lund, G. P. Niccolai, E. Lavoué, C. E. Hmelo-Silver, G. Gweon, and M. Baker (Eds.), International Society of the Learning Sciences, Lyon, France, 472–479.

[17] Sue Fitzgerald, Gary Lewandowski, Renée McCauley, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: Finding, Fixing and Flailing, a Multi-Institutional Study of Novice Debuggers. *Computer Science Education* 18, 2 (June 2008), 93–116. DOI: https://doi.org/10.1080/08993400802114508

[18] Virginia J. Flood, David DeLiema, Benedikt W. Harrer, and Dor Abrahamson. 2018. Enskilment in the Digital Age: The Interactional Work of Learning to Debug. In *Proceedings of International Conference of the Learning Sciences,* 3, 1405–1406.

[19] Virginia J. Flood, Christine Wang, and Michael Sheridan. 2022. Embodied Responsive Teaching for Supporting Computational Thinking in Early Childhood. In *Proceedings of the International Collaboration Towards Education Innovation for all: Overarching Research, Development, and Practices*, 855–862.

[20] Alexandra Gendreau Chakarov, Quentin Biddy, Colin Hennessy Elliott, and Mimi Recker. 2021. The Data Sensor Hub (DaSH): A Physical Computing System to Support Middle School Inquiry Science Instruction.*Sensors* 21, 18 (Sept. 2021), 6243. DOI: https://doi.org/10.3390/s21186243

[21] Jean M. Griffin. 2016. Learning by Taking Apart: Deconstructing Code by Reading, Tracing, and Debugging. In *Proceedings of the 17th Annual Conference on Information Technology Education.* ACM, Boston, MA, USA, 148–153. DOI: https://doi.org/10.1145/2978192.2978231

[22] Paulina Haduong and Karen Brennan. 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 (SIGCSE '19).* 1095–1101. DOI: https://doi.org/10.1145/3287324.3287479

[23] Rogers Hall and Reed Stevens. 2016. Developing Approaches to Interaction Analysis of Knowledge in Use. In *Knowledge and Interaction: A Synthetic Agenda for the Learning Sciences.* Andrea A. DiSessa, Mariana Levin, and Nathaniel J. S. Brown (Eds.), Routledge, New York, London, 72–108.

[24] Colin Hennessy Elliott, Alexandra Gendreau Chakarov, Jeffrey B. Bush, Jessie Nixon, and Mimi Recker. 2023. Toward a Debugging Pedagogy: Helping Students Learn to Get Unstuck with Physical Computing Systems. *ILS* 124, 1/2 (Feb. 2023), 1–24. DOI: https://doi.org/10.1108/ILS-03-2022-0051

[25] Patricio G. Herbst. 2003. Using Novel Tasks in Teaching Mathematics: Three Tensions Affecting the Work of the Teacher. *American Educational Research Journal* 40, 1 (Jan. 2003), 197–238. DOI: https://doi.org/10.3102/00028312040001197

[26] Gayithri Jayathirtha, Deborah Fields, and Yasmin B. Kafai. 2020. Pair Debugging of Electronic Textiles Projects: Analyzing Think-Aloud Protocols for High School Students' Strategies and Practices while Problem Solving. In *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS)*. M. Gresalfi and I. S. Horn (Eds.). International Society of the Learning Science, Nashville Tennessee, 1047–1054.

[27] Gayithri Jayathirtha, Deborah Fields, and Yasmin B. Kafai. Computational Concepts, Practices and Collaboration in High School Students' Debugging Electronic Textiles Projects. In *Proceedings of the International Conference on Computational Thinking Education 2018*. The Education University of Hong Kong, 27–32.

[28] Gayithri Jayathirtha and Yasmin B. Kafai. 2020. Interactive Stitch Sampler: A Synthesis of a Decade of Research on Using Electronic Textiles to Answer the Who, Where, How, and What for K–12 Computer Science Education. *ACM Transactions on Computing Education* 20, 4 (Nov. 2020), 1–29. DOI: https://doi.org/10.1145/3418299

[29] Brigitte Jordan and Austin Henderson. 1995. Interaction Analysis: Foundations and Practice. *Journal of the Learning Sciences* 4, 1 (Jan. 1995), 39–103. DOI: https://doi.org/10.1207/s15327809jls0401_2

[30] Yasmin B. Kafai, David DeLiema, Deborah A. Fields, Gary Lewandowski, and Colleen Lewis. 2019. Rethinking Debugging as Productive Failure for CS Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis, MN, USA, 169–170. DOI: https://doi.org/10.1145/3287324.3287333

[31] Yasmin Kafai, Deborah Fields, and Kristin Searle. 2014. Electronic Textiles as Disruptive Designs: Supporting and Challenging Maker Activities in Schools. *Harvard Educational Review* 84, 4 (Dec. 2014), 532–556. DOI: https://doi.org/10.17763/haer.84.4.46m7372370214783

[32] Yasmin Kafai, Nicole Hutchins, Caitlin Snyder, Karen Brennan, Paulina Haduong, Kayla DesPortes, David DeLiema, Oia Walker-van Aalst, Virginia Flood, Morgan Fong, Deborah Fields, Melissa Gresalfi, Corey Brady, Selena Steinberg, Diana Franklin, Donna Eatinger, Merijke Coenraad, Jen Palmer, David Weintrop, Michelle Wilkerson, Collette Roberto, and Nicole Bulalacao. 2020. Turning Bugs into Learning Opportunities: Understanding Debugging Processes, Perspectives, and Pedagogies. In *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS)*, Vol. 1. M. Gresalfi and I. S. Horn (Eds.), 374–381.

[33] Manu Kapur and Katerine Bielaczyc. 2012. Designing for Productive Failure. *Journal of the Learning Sciences* 21, 1 (Jan. 2012), 45–83. DOI: https://doi.org/10.1080/10508406.2011.591717

[34] ChanMin Kim, Jiangmei Yuan, Lucas Vasconcelos, Minyoung Shin, and Roger B. Hill. 2018. Debugging during Block-Based Programming. *Instructional Science* 46, 5 (Oct. 2018), 767–787. DOI: https://doi.org/10.1007/s11251-018-9453-5

[35] Amy J. Ko, Thomas D. LaToza, Stephen Hull, Ellen A. Ko, William Kwok, Jane Quichocho, Harshitha Akkaraju, and Rishin Pandit. 2019. Teaching Explicit Programming Strategies to Adolescents. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis, MN, USA, 469–475. DOI: https://doi.org/10.1145/3287324.3287371

[36] Amy J. Ko and Brad A. Myers. 2003. Development and Evaluation of a Model of Programming Errors. In *Proceedings of the IEEE Symposium on Human Centric Computing Languages and Environments*. IEEE, Auckland, New Zealand, 7–14. DOI: https://doi.org/10.1109/HCC.2003.1260196

[37] Emine Kutay and Diler Oner. 2022. Coding with Minecraft: The Development of Middle School Students' Computational Thinking. *ACM Transactions on Computing Education* 22, 2 (June 2022), 1–19. DOI: https://doi.org/10.1145/3471573

[38] Michael Lachney, Aman Yadav, Matt Drazin, Madison C. Allen, and William Babbitt. 2021. Culturally Responsive Debugging: A Method to Support Cultural Experts' Early Engagement with Code. *TechTrends* 65, 5 (Sept. 2021), 771–784. DOI: https://doi.org/10.1007/s11528-021-00618-4

[39] Victor R. Lee and Deborah A. Fields. 2017. A Rubric for Describing Competences in the Areas of Circuitry, Computation, and Crafting after a Course Using E-Textiles. *International Journal of Information and Learning Technology* 34, 5 (Nov. 2017), 372–384. DOI: https://doi.org/10.1108/IJILT-06-2017-0048

[40] Chen Li, Emily Chan, Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2019. Towards a Framework for Teaching Debugging. In *Proceedings of the 21st Australasian Computing Education Conference (ACE '19)*. ACM Press, Sydney, NSW, Australia, 79–86. DOI: https://doi.org/10.1145/3286960.3286970

[41] Neomi Liberman, Yifat Ben-David Kolikant, and Catriel Beeri. 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 (Sept. 2012), 257–283. DOI: https://doi.org/10.1080/08993408.2012.721663

[42] Eve Manz. 2015. Resistance and the Development of Scientific Practice: Designing the Mangle into Science Instruction. *Cognition and Instruction* 33, 2 (Apr. 2015), 89–124. DOI: https://doi.org/10.1080/07370008.2014.1000490

[43] Renée McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: A Review of the Literature from an Educational Perspective. *Computer Science Education* 18, 2 (June 2008), 67–92. DOI: https://doi.org/10.1080/08993400802114581

[44] Tilman Michaeli and Ralf Romeike. 2019. Current Status and Perspectives of Debugging in the K12 Classroom: A Qualitative Study. In *Proceedings of the 2019 IEEE Global Engineering Education Conference (EDUCON '19)*. IEEE, Dubai, United Arab Emirates, 1030–1038. DOI: https://doi.org/10.1109/EDUCON.2019.8725282

[45] Matthew B. Miles, A. M. Huberman, and Johnny Saldaña. 2020. *Qualitative Data Analysis: A Methods Sourcebook* (4th. ed.). SAGE, Los Angeles.

[46] Emily Miller, Eve Manz, Rosemary Russ, David Stroupe, and Leema Berland. 2018. Addressing the Epistemic Elephant in the Room: Epistemic Agency and the Next Generation Science Standards. *Journal of Research in Science Teaching* 55, 7 (Sept. 2018), 1053–1075. DOI: https://doi.org/10.1002/tea.21459

[47] Jessie Nixon, Colin Hennessy Elliott, Michael Schneider, Jeffrey B. Bush, Srinjita Bhaduri, and Mimi Recker. 2023. Teachers' Learning to Support Students during Science Inquiry: Managing Student Uncertainty in a Debugging Context. In *Proceedings of the 17th International Conference of the Learning Sciences (ICLS'23)*. P. Blikstein, J. Van Aalst, R. Kizito, and K. Brennan (Eds.), International Society of the Learning Sciences, 601–608. DOI: https://doi.org/10.22318/icls2023.247673

[48] Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York.

[49] Roy D. Pea, Elliot Soloway, and Jim C. Spohrer. 1987. The Buggy Path to the Development of Programming Expertise. *Focus on Learning Problems in Mathematics* 9, 1 (1987), 5–30.

[50] D. N. Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. 1986. Conditions of Learning in Novice Programmers. *Journal of Educational Computing Research* 2, 1 (Feb. 1986), 37–55. DOI: https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL

[51] Thomas M. Philip. 2019. Principled Improvisation to Support Novice Teacher Learning. *Teachers College Record* 121, 6 (June 2019), 1–32. DOI: https://doi.org/10.1177/016146811912100607

[52] Chris Proctor, Maxwell Bigman, and Paulo Blikstein. 2019. Defining and Designing Computer Science Education in a K12 Public School District. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, MN, USA, 314–320. DOI: https://doi.org/10.1145/3287324.3287440

[53] John Pryor and Barbara Crossouard. 2008. A Socio-Cultural Theorisation of Formative Assessment. *Oxford Review of Education* 34, 1 (Feb. 2008), 1–20. DOI: https://doi.org/10.1080/03054980701476386

[54] Josh Radinsky. 2008. Students' Roles in Group-Work with Visual Data: A Site of Science Learning. *Cognition and Instruction* 26, 2 (Apr. 2008), 145–194. DOI: https://doi.org/10.1080/07370000801980779

[55] Brian J. Reiser. 2014. *Designing Coherent Storylines Aligned with NGSS for the K-12 References 5 Classroom*. Presentation at the National Science Education Leadership Association Meeting. Boston, MA. Retrieved from https://www.academia.edu/6884962/Designing_Coherent_Storylines_Aligned_with_NGSS_for_the_K_12_Classroom

[56] Zachary D. Ryan and David DeLiema. 2023. Reflections on Sustained Debugging Support: Conjecture Mapping as a Point of Departure for Instructor Feedback on Design. *Instructional Science* 51, 6 (Dec. 2023), 1043–1078. DOI: https://doi.org/10.1007/s11251-023-09629-5

[57] R. Keith Sawyer. 2004. Creative Teaching: Collaborative Discussion as Disciplined Improvisation. *Educational Researcher* 33, 2 (Mar. 2004), 12–20. DOI: https://doi.org/10.3102/0013189X033002012

[58] Sue Sentance, Jane Waite, Lucy Yeomans, and Emily MacLeod. 2017. Teaching with Physical Computing Devices: The BBC micro:bit Initiative. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM, Nijmegen, Netherlands, 87–96. DOI: https://doi.org/10.1145/3137065.3137083

[59] Lucille Alice Suchman. 1987. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge, New York.

[60] Iris Vessey. 1985. Expertise in Debugging Computer Programs: A Process Analysis. *International Journal of Man-Machine Studies* 23, 5 (Nov. 1985), 459–494. DOI: https://doi.org/10.1016/S0020-7373(85)80054-7

[61] Lev Vygotsky. 1978. *Mind in Society: The Development of Higher Psychological Processes* (Revised ed.). Harvard University Press, Cambridge.

[62] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology* 25, 1 (Feb. 2016), 127–147. DOI: https://doi.org/10.1007/s10956-015-9581-5

[63] Aman Yadav, Sarah Gretter, Susanne Hambrusch, and Phil Sands. 2016. Expanding Computer Science Education in Schools: Understanding Teacher Experiences and Challenges. *Computer Science Education* 26, 4 (Dec. 2016), 235–254. DOI: https://doi.org/10.1080/08993408.2016.1257418

Appendix A

A Summary of Debugging Moments

Teacher	Label in Continuum	School Year	Length of Moment	Initiated by teacher (T) or student (S)	Bug Characterization[a]
Karla	K.1	Fall 2021	75 seconds	S	Transferring program from MakeCode to Micro:bit (HW+SW)
	K.2	Fall 2021	57 seconds	S	Program did not start until initiated (HW+SW)
	K.3	Fall 2021	140 seconds	S	Cannot determine from data
	K.4	Fall 2021	61 seconds	T	Blocks not in scope (HW)
	K.5	Fall 2021	225 seconds	S	Transferring program from MakeCode to Micro:bit (HW+SW)
Dave	D.1	Fall 2021	251 seconds in two parts (120 + 131)	S	Cannot determine from data
	D.2	Fall 2021	143 seconds in two parts (68 + 71)	S	Cannot determine from data
	D.3	Fall 2021	40 seconds	S	Assembly errors (HW)
	D.4	Fall 2021	105 seconds	T	Conditional Logic (SW)
	D.5	Fall 2021	126 seconds	S	Conditional Logic (SW)
Natalie	N.1	Spr 2022	59 seconds	S	Syntax issue (SW)
	N.2	Spr 2022	106 seconds	S	Conditional logic (SW)
	N.3	Spr 2022	238 seconds	T	Conditional Logic (SW)
	N.4	Spr 2022	135 seconds	S	Conditional Logic (SW)
	N.5	Spr 2022	348 seconds	T	Conditional Logic (SW)
Uma	U.1	Fall 2019	77 seconds	S	Transferring program from MakeCode to Micro:bit (HW+SW)
	U.2	Fall 2019	167 seconds	T	Assembly errors (HW) & Conditional Logic (SW)
	U.3	Fall 2019	54 seconds	S	Transferring program from MakeCode to Micro:bit (HW+SW)
	U.4	Fall 2019	67 seconds	T	Conditional Logic (SW)
Gabrielle[b]	G.1	Fall 2019	169 seconds	T	Cannot determine from data
	G.2	Fall 2019	70 seconds	S	Assembly setup error (HW)
	G.3	Fall 2019	113 seconds	S	Sensor Values (SW)
	G.4	Fall 2019	170 seconds	S	Sensor Values (SW)

Notes

[a]Bug characterizations based on our previous [24] characterizations of common errors in hardware (HW), software (SW), and combination of hardware and software (HW+SW).

[b]Gabrielle was the only participant who had previously implemented the sensor immersion unit

Received 30 June 2022; revised 13 March 2024; accepted 13 June 2024