Chapter 16
Teaching Computational Thinking with Electronic Textiles: Modeling Iterative Practices and Supporting Personal Projects in *Exploring Computer Science*

Deborah A. Fields, Debora Lui and Yasmin B. Kafai

Abstract Iterative design is an important aspect of computational thinking in which students learn to face challenges and persevere in fixing them. Yet we know little of how school teachers can support students in using iterative practices. In this chapter, we consider the teaching practices of two experienced computer science teachers who implemented a new 8-week long unit on making electronic textiles in their classrooms. Electronic textiles are sewn, programmable circuits with sensors and actuators on personal artifacts that provide various opportunities to learn through mistakes. Through analyses of observations and interviews with students and teachers who implemented the unit, we identified several teaching practices that supported values of iteration, revision, and working through mistakes. These included teachers modeling their own processes and mistakes in making projects, teachers modeling students' mistakes to the wider class, and supporting personalized projects that resulted in unique "bugs" or challenges for each student. After sharing examples of these practices, we consider student and teacher reflections on the ways that mistakes and iteration supported student learning.

Keywords Computational thinking · Iteration · Computer science education · Electronic textiles · Teaching practices

D. A. Fields (✉)
Utah State University, 2830 Old Main Hill, Logan, UT 84322, USA
e-mail: deborah.fields@usu.edu

D. Lui · Y. B. Kafai
University of Pennsylvania, 3700 Walnut Street, Philadelphia, PA 19104, USA
e-mail: deblui@upenn.edu

Y. B. Kafai
e-mail: kafai@upenn.edu

S.-C. Kong and H. Abelson (eds.), *Computational Thinking Education*,
https://doi.org/10.1007/978-981-13-6528-7_16

16.1 Introduction

The introduction of computational thinking into the K-12 curriculum has become a global effort. Computational thinking (CT) was defined by Wing (2006) as a way of approaching and conceptualizing problems, which draws upon concepts fundamental to computer science such as abstraction, recursion, or algorithms. Early work in this area primarily focused on defining computational thinking, specifically its cognitive and educational implications as well as highlighting existing contexts for teaching computational thinking (e.g., NRC, 2011). While much subsequent work has focused on the development of different environments and tools for CT, as well as curricular initiatives in the K-12 environment, there is growing need for more empirical work situated in actual classroom environments (Grover & Pea, 2013).

Iterative design is an important aspect of computational thinking that involves engaging in an adaptive process of design and implementation where students learn to face challenges and persevere in fixing them (Brennan & Resnick, 2012). Yet one glaring absence in the work on iteration in computational thinking is a lack of understanding exactly how *teachers* can support such CT practices in their classrooms (Barr & Stephenson, 2011). Thus far, most studies focused on CT tools and environments had researchers themselves implement projects or were situated in out-of-school contexts where youth voluntarily engaged on topics of their own choosing (e.g., Grover, Pea, & Cooper, 2015; Denner, Werner, & Ortiz, 2012). While these studies provide important insights about the feasibility of engaging students in CT, they could not address the critical issue of how computer science teachers, dealing with large class sizes and curricular restrictions, can integrate CT into their classroom activities by connecting technology, content, and pedagogy (Mishra & Kohler, 2006).

In this paper, we focus on how two high school teachers supported iterative practice as a core CT practice during their implementation of an eight-week (~40 h) electronic textiles unit within their classrooms during the year-long *Exploring Computer Science* (ECS) curriculum (Goode, Margolis, & Chapman, 2014). Electronic textiles (e-textiles), or fabric-based computing, incorporate basic electronics such as microcontrollers, actuators and sensors with textiles, conductive thread and similar "soft" materials (see Buechley, Peppler, Eisenberg, & Kafai, 2013a). Two researchers observed the daily implementation of the curriculum, documenting classroom activities and interactions in extensive field notes, video recordings and photos of students' work. The following research question guided our analysis "What kind of teaching strategies did the teachers employ to support students' iterative practice during the e-textiles unit?" Our discussion focuses on the teachers' modeling and personalization strategies to make iteration accessible in students' work, particularly through classroom practices that support iteration.

16.2 Background

While computational thinking is related to the creation of code, it is important to note how understanding programming is not the same thing as CT itself (Wing, 2006). As Wing (2006) states, "[t]hinking like a computer scientist means more than being able to program a computer." In other words, it involves particular kinds of approaches to problems that exist in the world (not just on the screen). In terms of teaching programming, considerable research has focused on *content,* drawing attention to the ways in which particular programming concepts and practices, such as loops and debugging, can be taught within classrooms (e.g., Soloway & Spohrer, 1989). Here, research is driven by the need to recognize what concepts and practices are difficult to learn and how to scaffold students' learning. More recent efforts have focused on *context*, highlighting different kinds of projects and applications in which learning programming can occur, whether in game design, robotics, creating apps, or constructing wearables such as e-textiles (e.g., Kafai & Burke, 2014). Here, efforts are driven by the recognition that teaching and learning programming need to be contextualized in ways that engage students' existing interests.

Because teaching computational thinking is newer than teaching programming, research has generally focused more broadly on conceptual or hypothetical contexts (Grover & Pea, 2013). One goal has been to define the actual nature of computational thinking in terms of cognition and its relationship to existing disciplines such as mathematics and engineering (NRC, 2011). Other work has focused on developing CT-focused curricula for K-12 contexts such as *AP Computer Science Principles* (e.g., Guzdial, 2016), *Exploring Computer Science* (Margolis & Goode, 2016), or in science and mathematics classes (e.g., Tofel-Grehl et al., 2017; Weintrop et al., 2016). Finally, researchers have identified the importance of developing particular environments and tools for supporting CT, often overlapping with those that teach programming (e.g., graphical programming interfaces, digital and tangible computational construction kits).

While all this work focuses on the potential or need to bring CT into education, what is missing are studies of how teachers actually implement these ideas in their classrooms and the particular ways in which computational thinking tools, content, and pedagogy intersect. Approaching this effort, Mishra and Kohler (2006) described *technological pedagogical content knowledge*, relating education with digital technology more broadly, but not specifically with computing or computer science. Instead, deeper understanding on computer science teaching is still in early stages compared to other disciplines such as math and sciences. For the most part, research on actual computer science teaching, with a focus on CT, has focused on pre-service teachers and ways to integrate CT in classrooms (e.g., Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014). Case studies have been developed to examine the strategies used by teachers to address CT in their classrooms (Griffin, Pirman, & Gray, 2016). The area that overlaps most with CT is focused on algorithmic thinking (Ragonis, 2012). A few scholars such as Margolis, Goode, and Ryoo (2015) have

studied CS teachers' pedagogical practices, such as inquiry-based strategies, but not specifically how pedagogy connected with CT or CS concepts.

Our work contributes to this newly emerging body of *computational pedagogical content knowledge* by examining how experienced computer science teachers teach CT using electronic textiles. Early studies of e-textiles focused on broadening participation in areas of computing and engineering by reshaping students' perspectives of and interests in those fields (e.g., Buchholz, Shively, Peppler, & Wohlwend, 2014; Kafai, Fields, & Searle, 2014a). One study (Kafai et al., 2014a) identified several CT concepts, practices and perspectives that students learned while making an e-textiles human sensor project—a precursor to one of the projects in the curriculum discussed in this paper. For example, this project included concepts such as sequencing, since students had to properly sequence code in order to coordinate behavior of the sensors and lights in their project, as well as practices like remixing and reusing code, since students had to modify a sample program in order to accommodate different circuit diagrams, sensor types, and intended behaviors. However, while that study identified ways that making e-textiles can support CT, the unit was taught by researchers and pedagogy was not a focus of the study.

In this chapter, we focus on how e-textiles and pedagogy can be used to support the essential CT practice of *iterative design*, looking at it as a core computational thinking practice alongside pedagogy. There are many areas of computational thinking to address, so why did we choose iterative design? This choice stems from an analysis of teaching practices that supported equity and broadening participation in two ECS classrooms that piloted an e-textile unit (Fields, Kafai, Nakajima, Goode, & Margolis, 2018a). We noted in our analysis that teachers' support for iteration happened at multiple levels: whole class modeling, individual student work, collaborative student work, and in the design of the classroom learning environment itself. Thus iterative design stood out as an area of equitable teaching practice that simultaneously broadened participation in computer science while introducing and reinforcing an important practice of computational thinking.

Within the world of software design, iteration—or the process of continual repetition and revision—is essential for the completion and refinement of different algorithms and programs and is therefore considered a key *conceptual idea* within CT, alongside conditional logic, abstraction, and algorithms (Brennan & Resnick, 2012; Grover & Pea, 2013; Wing, 2006). However, the act of iteration itself is also thought of as an important *practice* within CT in and of itself, when actually engaging with computational problems and projects. As outlined by Brennan and Resnick (2012), iterative design is defined as the cycle of prototyping, testing, and revision, that requires students to engage in a continually "adaptive process," throughout the course of creating a computational artifact, where one's goals "might change in response to approaching a solution in small steps" (p. 7). Iterative design is also a key practice highlighted in new secondary school computer science curricula such as the AP Computer Science Principles course, which reinforces the importance of iterative software development as "essential knowledge" in many different learning objectives in the course (e.g., CollegeBoard, 2016: principles EK 1.1.1B, 3.1.1A, 4.1.1A, 4.1.1D, 4.1.2G, 5.1.2A, 5.1.3C, 5.5.1J). From this perspective, iteration is not only a

concept that can be applied into the body of a computer program itself, but can also refer to the purposefully incremental process of creating a computational artifact. Here, it is not only the act of iteration that matters, but an awareness and explicit acknowledgment of its importance in tackling problems in a systematic way—basically, learning how to think in an iterative way about real-world issues.

Iterative practices are also important within the field of engineering where they involve engagement with trial-and-error processes, along with revision and refinement of ideas over time (Barr & Stephenson, 2011; Lee et al., 2011). Within engineering education, these processes of iteration and revisions are more formally structured into classroom practice through the model of the engineering design process, which highlights the steps of prototyping, testing, and redesign (Tayal, 2013). From this perspective, the practice of iterative design should be considered something to be supported within both CT curricula and contexts. However, because CT-focused curricula and pedagogy are newer, there remains a critical need to highlight how iterative design as a practice can be supported through pedagogical interventions.

Using e-textiles affords different opportunities to observe teaching strategies to support iterative design because they (1) integrate CT within both programming (i.e., software design) and engineering (i.e., circuit design, physical craft) and can illustrate how teachers make connections between these contexts; (2) are hybrid nature in nature (i.e., as textual code on the screen and as physical circuits on the textile) and can make visible how teachers navigate between different modalities; and (3) allow for creative expression and aesthetics through personalized projects and can demonstrate how teachers respond to and are supportive of distinct student interests. Focusing on two classrooms from the *Exploring Computer Science* (ECS) program (Goode et al., 2014), we examined what strategies these experienced ECS teachers used in their implementation of the new e-textiles curriculum unit (Fields, Lui, & Kafai, 2017; Fields et al., 2018a, b). In this chapter we focus on strategies that support *iteration* as a key computational thinking process that can be difficult to implement in classrooms.

16.3 Methods

16.3.1 Context

Our e-textiles unit is embedded within the *Exploring Computer Science* (ECS) initiative, which comprises a one-year introductory computer science curriculum with a 2-year professional development sequence. The curriculum consists of six units: Human–Computer Interaction, Problem-Solving, Web Design, Introduction to Programming (Scratch), Computing and Data Analysis, and Robotics (Lego Mindstorms) (Goode & Margolis, 2011). The instructional design of the curriculum adopts inquiry-based teaching practices so that all students are given opportunities to explore and design investigations, think critically and test solutions, and solve real problems.

ECS has successfully increased diversity to representative rates in Los Angeles and has subsequently scaled nationwide to other large urban districts and regions, now with over 500 teachers nationwide.

Within this successfully implemented, inquiry-based curriculum, we noted an opportunity to broaden the range of computer science activities by including e-textiles. The curriculum unit was co-developed by e-textiles and ECS experts to combine best practices of teaching and crafting e-textiles based on a constructionist philosophy alongside ECS principles, style, and writing. The curriculum contains big ideas and recommended lesson plans, with much room for teachers to interpret and bring in their own style. A final version of the curriculum can be found at http://exploringcs.org/e-textiles.

The ECS e-textiles unit implemented for this study consisted of six projects, each increasing in difficulty and creative freedom, that introduced concepts and skills including conductive sewing and sensor design; simple, parallel, and computational circuits (independently programmable); programming sequences, loops, conditionals, and Boolean logic; and data from various inputs (switches and sensors). The projects were as follows: (1) a paper-card using a simple circuit, (2) a "stitch-card" with one LED sewn as a simple circuit, (3) a wristband with three LEDs in parallel, (4) a felt project using a preprogrammed LilyTiny microcontroller and 3–4 LEDs, (5) a classroom-wide mural project where pairs of students created portions that each incorporated two switches to computationally create four lighting patterns, and (6) a "human sensor" project that used two aluminum foil conductive patches that when squeezed generated a range of data to be used as conditions for lighting effects. Student artifacts included stuffed animals, paper cranes, and wearable shirts or hoodies, all augmented with the sensors and actuators.

In Spring 2016 two high school teachers, each with 8–12 years of computer science classroom teaching experience, piloted the e-textiles unit in their *ECS* classes in two large public secondary schools in a major city in the western United States. Both schools had socioeconomically disadvantaged students (59–89% of students at each school) with ethnically non-dominant populations (i.e., the majority of the students at each school include African American, Hispanic/Latino, or southeast Asian students).

16.3.2 Data Collection and Analysis

The study is part of a larger design-based implementation research study (Penuel, Fishman, Cheng, & Sabelli, 2011) where the goal is to develop and revise an e-textiles unit over the course of 3 years, attend to problems of practice in the classroom, develop better theories of pedagogy related to making and computing, and support classrooms in sustainable changes as they bring making to computer science. This paper reports on the first year of the study, where two teachers implemented the curriculum for the first time. Two researchers gathered data focused on teacher practice in the classroom, visiting each class equally, four days a week (about 8 weeks, with

interruptions from holidays, testing, and other school obligations). The researchers documented teaching with detailed field notes, in-class video and audio recordings, and pictures/videos of student work, supplemented by three interviews with the teachers before, during, and after the unit, and brief focus group interviews with students at the end of the unit.

The analysis of field notes involved constant comparative analysis (see Charmaz, 2011) to (1) identify computational thinking practices exhibited during the e-textiles unit, and then (2) compare these with the larger corpus of computational thinking practices identified in the AP Computer Science Principles curriculum (see Fields et al., 2017). Through this process, iteration stood out a key area of learning. Then the team re-coded the data to find all of the teaching practices in this area. Finally, the team compared findings from observational data with the interviews from teachers and students to see whether these practices came up from participants' perspectives and to understand these two areas in greater depth.

16.4 Findings: Contexts for Iteration: Creating a Classroom Culture Valuing Mistakes and Revisions

Within the e-textiles unit, students engaged with iteration at many stages: prototyping, testing, and revising designs while tackling bugs and problems that arose in the process. This was evident through the changes that students made in their projects, including improvements in circuit diagrams, changes in and expansions of code, and visible alterations in the physical projects themselves. In fact, iterating on project ideas and implementation was one of the most frequent things we coded across the data. Earlier work in e-textiles has documented similar changes in student design (Fields, Kafai, & Searle, 2012; Kafai et al., 2014b). The focus of our findings here is on how the teachers supported a culture of iteration and refinement in their classrooms. What practices did they use to create an environment where sharing about mistakes and seeing them as a means of learning was valued? Furthermore, how did this culture support students' awareness and acknowledgement of iteration as a key perspective that could be used in service of creating a computational artifact? Below we outline three main areas of teaching practices that helped to develop a classroom culture of iteration: teachers' modeling of their own mistakes and teachers' modeling of students' mistakes, all with a focus on students' personal designs.

16.4.1 Teachers Modeling Their Own Mistakes

One key teaching practice involved teachers promoting *their own mistakes, errors, and less-than-perfect projects* in front of the classroom. When introducing a new project, for instance, Ben or Angela would show their own sample creations (which

were made during teacher professional development for the unit). This not only served to give students ideas but also allowed the teachers to showcase their own experience of revision and iteration, and coach students on tips for dealing with this process. Consider the way Angela shared her work in her introduction to the LilyTiny project in class, as highlighted within our field notes

> So let me tell you a little story. When I was working on one of my projects, I didn't think I needed a plan. "I'll just do this," I thought.' Angela went on to explain that she worked for two days on her project and then eventually had to take most of it apart because it didn't work. 'If you don't plan, it's going to take you more time to take things out and fix it than it would to do it right the first time. So you're going to draw it all out, and you'll use that blueprint and then you'll have your little map. (160405 field notes[1]).

Here Angela told a self-deprecating story of her process of creating her project. She highlighted how *not* having a predetermined plan for her e-textiles project—specifically, mapping out circuitry connections beforehand—meant that she ended up making numerous mistakes while crafting and eventually had to redo her entire project. She framed the circuit diagram as a way of creating a well-thought out plan for implementation, a skill which is helpful not only with regard to crafting but also within the context of creating a complex computational artifact that encompasses multiple modes (i.e., sewing, circuitry, programming). It serves as both a time-saving measure to prevent costly mistake fixing, but also to help students structure their construction time more efficiently. Note that the knowledge Angela shared was very pragmatic; she did not ask students to trust her on her authority alone (i.e., "have a plan because I said so") but rather because of her personal experience.

The other teacher, Ben, similarly highlighted his own process of dealing with mistakes when using his own project to aid in teaching. During a programming lesson, Ben shared his personal project code as an exemplar and sample for students to remix while using a preassembled e-textiles circuit board (the LilyPad Protosnap)

> Ben: As an introduction, "everyone please open up my.pdf called Function Code. And I want pairs to inspect the code and discuss what the code is going to do." Two students who were absent the prior day pointed to the switch and said that Number 2 should turn on… They seemed confused. At that moment, Ben realized his error—the code that he shared was something he wrote up for his own (extra) project that he is making that will involve switches and buzzers —his variables refer to the wrong ports on the Protosnap microcontroller. (160524 field notes).

Ben went on to explain this mistake to the class: "My apologies. I was messing around with the buzzer last night [on my own project]," and then explained how the connections were different from the LilyPad Protosnap boards students were using. However, rather than starting over on the task, Ben highlighted his error to students, inviting them to participate in how it could be fixed so that the code matched with the boards (160524 field notes).

---

[1]We use double quotation marks (" ") to show exact words of participations and single quotation marks (' ') to show paraphrased words that occur most often in field notes where conversations were typed in the moment rather than audio recorded and transcribed.

In modeling their own imperfect processes of creation and addressing their mistakes, the teachers, Angela and Ben, thereby promoted a classroom culture of iterative practice, valuing process over product. While everyone was encouraged to make full, working projects, the teachers stressed that the actual experience of creation and learning would most likely include moments of failure, and subsequent revisions and iterations. Students were encouraged to think that it was okay not to be perfect the first time (or the second, third, fourth) they did something. Perfection, in these cases, could prevent students from moving forward in their learning.

16.4.2 Teachers Modeling Students' Mistakes

Just as the teachers showed their own projects and processes in front of the entire class, they also showcased students' challenges, mistakes, and in-process projects in order to promote the practice of iteration and revision. For instance, Angela added a journal question after the completion of the wristband activity that solicited challenges that students had faced: "Think about this week's project, what was the biggest challenge? If you had no challenges, what tips do you have for people who may be struggling?" (160422 field notes). After students had some minutes to think and to write, she invited various students to share out what they had written. Numerous students shared advice such as "plan more," echoing Angela's lesson from above regarding the importance of creating clear circuit diagrams as blueprints in help in constructing a functional e-textile artifact. Students also mentioned other issues that spanned across the multiple domains of e-textiles work. For instance, another student brought up the polarity issue of "mixing up the positive and negative" when trying to create a working sewn circuit. In response, Angela invited students to share suggestions on how to avoid this issue; they suggested curling or twisting the positive and negative sides of the LED wires to look different, developing symbolic means to identify polarity.

Another way that the teachers modeled students' mistakes was through reflections between projects. For instance, after the wristband (project #3) was complete, Ben provided some constructive thoughts to his class

Ben: 'I want to talk about the [project] we just did. Because those bracelets look awesome, they look fantastic and you should be proud of yourselves. There were a couple of things that I saw that you could improve … I saw some sloppy stitching. Meaning that they were more than an inch. Some of them were not pulled completely tight. I know that some of you might not want to do the work of going in and out and in and out. But what might be a concern?'

Student: 'It would get caught in something.'

Ben: 'Yes, and concise means a little tighter.' (160418, field notes)

In this example, Ben went on to coach students about two other problems he saw in student work on the wristbands. Each time, he presented a problem then asked

students why they thought it might be an issue, allowing the students to share their expertise on these problems. In this way Ben also supported iterative design between projects, not just within a project. This suggests that having a *series* of projects provided more opportunities for iterative practices than just having a single project. Students could improve their techniques across projects by recalling this ongoing catalogue of mistakes and solutions, and essentially acting as problem-solving resources for one another. Beyond merely supporting *iterative activity*, the teaching strategies described above pushed students to explicitly consider how *iterative thinking* was an essential part of computational work. By asking students to continually reflect upon the challenges and issues they faced, both Angela and Ben were able to highlight the inherently adaptive, trial and error nature of creating a functional e-textile artifact. Students learned not only how to recognize potential problems, but how to continually address these through more effective planning and ongoing implementation. From this standpoint, students became more cognizant of the powerful role that iteration could play in developing their own increasingly complex projects over time.

16.4.3 Supporting Personalized Design to Facilitate Iterative Practice

What inspired these practices of revealing and sharing mistakes, thus reinforcing process over product, and supporting iteration as an activity and a perspective? One thing that both teachers noted in their interviews with us was that doing the projects themselves helped them understand what students were going through. As Angela expressed, "A lot of times, we don't do what we're asking kids to do. It was great doing the projects in the [professional development sessions] because it helped me anticipate questions that might come up in class and think of ways to address them" (160609, interview). Creating projects helped the teachers themselves to reflect on their own processes of making them (including their own experience of trial and error, revision, and iteration) and provided a base of stories and examples to share with students. Thus one of the most important underlying aspects of the classroom culture was the teachers' *emphasis on their own as well as students' personalized projects*.

How did the teachers' promotion of personalized projects promote iterative thinking and design? Students were tasked to create projects within predetermined constraints throughout the unit (e.g., a light-up wristband, an interactive felt banner). However, both Angela and Ben also actively encouraged students to develop their own personal ideas through these assignments. This can be illustrated, in particular, through the students' final human sensor project, where a fabric object was modified to include four to five LEDs, which could be triggered by readings from conductive foil patches into at least four customized light pattern functions. From the start, the teachers actively encouraged personalization by allowing students to either bring or

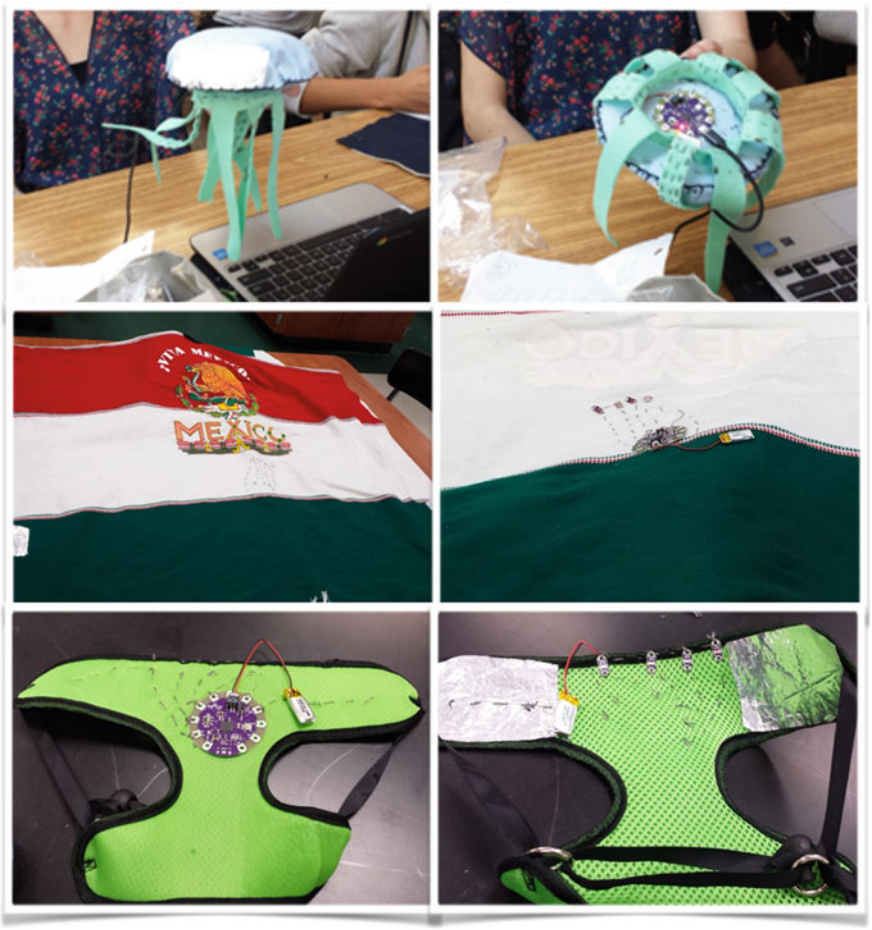

Fig. 16.1 Human sensor projects by students (top to bottom): Bridget's jellyfish (top and bottom views), Mateo's Viva Mexico poncho (front and back), and Peter's dog harness (top and bottom)

create their own personal objects for modification. This ranged from a "Viva Mexico" poncho, to a dog walking harness, to a stuffed jellyfish made of fabric (see Fig. 16.1). Further, teachers encouraged the students to customize the desired functionality based on their own interests and desires. For instance, some projects had a practical purpose (the walking harness that lit up when it was actually worn by a dog), while others' projects were more whimsical (the jellyfish, whose tentacles glowed when triggered through play and interaction).

By pushing this personalization within the context of the given project constraints, teachers inherently led students through an iterative process. One way this occurred was in developing unique circuit diagrams for the diverse objects. For instance, Angela had to assist students in thinking through the complex spatial dimensions

translating a two-dimensional blueprint to many different kinds of three-dimensional objects, for instance, a hat with an inside and outside surface or a stuffed animal with multiple overlapping surfaces. Another important context of iteration was in programming their objects. Students were all given basic "starter code" that included the functions for activating the conductive patches and a conditional statement with different outputs based on sensor readings. To match individual projects, students were required to make multiple changes. Ben, for instance, worked with his entire class to test how the different conductive patch sizes and users would shift the reading ranges that would need to be included within students' different programs.

Notably, this emphasis on original design ensured feelings of personal ownership over student projects, which in turn led to a willingness to persevere through mistakes and bugs. Because each project was distinct, students dealt with unique sets of challenges and issues. Troubleshooting therefore became a process of knowing how to isolate different problems, iteratively moving through different potential causes and solutions. While some students did become discouraged by this process, for others, the personal commitment to the project led them to push through the process, even beyond the teachers themselves, as demonstrated by the example from an observation below

> Ethan's project was finished but a glitch in the code caused two of the LEDs not to turn on in two of the four lighting pattern functions. The teacher and the researcher worked to help Ethan troubleshoot the project; Ethan expressed great frustration. Two other students came to try to assist. One said that she had a problem with delays in her code, so Ethan tried many different types of code fixes (added delays, copied and pasted code into a new function, re-typed the problematic function). No one could figure out the underlying problem. Unbothered at this point, Ethan finished by rewriting the code for the two problematic functions, designing simpler lighting functions, and it worked (160531, field notes).

From this standpoint, encouraging personalization not only creates natural opportunities for iterative design and troubleshooting, but also has the potential of motivating students to continually engage with and push through these processes. Considering that incremental and iterative work is so essential to computational thinking, the personalization promoted by the teachers and afforded by the medium of e-textiles therefore further enhances iteration.

16.5 Discussion

Our paper contributes to the emerging body of research on teaching practices of computational thinking that focused not just on programming but also on physical crafting and electronics. In our analysis we focused on a key aspect of computational thinking—iterative practices—that teachers addressed within the electronic textiles curriculum unit. The teachers illustrated how to support CT with personalized project creation involving large numbers of students (24 and 35), the restricted time constraints of class periods, and the temporary nature of classroom-based makerspaces where materials had to be put away every day. In the following sections, we dis-

cuss further aspects of teaching computational thinking that we saw instantiated in our study with the goal to make teaching CT more culturally relevant and equitable (Goode et al., 2014).

First, we draw attention to *how* teachers engaged with iterative practices: by not only modeling publicly their own mistakes but also those of their students so that others could learn from them. These combined practices created a classroom culture where process—improving, fixing, iterating on designs—was situated in the context of students' personal projects addressing their individual concerns while also sharing them with others. This approach to teaching computational thinking made iterative practices public and part of the larger classroom community while also distributing responsibilities. Through this approach teachers validated "process" alongside "product", highlighting iteration not only as an activity, but as a perspective which can help organize students' engagement with computational problems. In more recent efforts we are working to further enhance this approach by having students report and reflect on their mistakes in portfolios (see Lui, Jayathirtha, Fields, Shaw, & Kafai, 2018; Lui et al., in press; Fields, Shaw, & Kafai, 2018). Our plan is to have students document the challenges and revisions that come up in their iterative design processes by taking pictures of mistakes, marking errors in versions of code, or showing changes in the development of their circuit diagrams. This will add a formal element of reflection on iterative design to the pedagogy that the teachers have already implemented in their classrooms.

Second, we point out that by making computational thinking public and part of the larger classroom culture, the teachers also addressed another critical element relevant to supporting equitable and inquiry-based teaching: creating an audience for e-textiles projects that highlights their usability. For instance, by sharing their projects and discussing multiple users of their projects (including one's spouse) the teachers brought out the broader usability of projects: students' projects as well as their teachers' projects could have relevance outside the classroom. These features are important because audience and participation are key areas of supporting design though they have only recently been recognized as relevant in the area of computational thinking by highlighting computational participation (Kafai & Burke, 2014). Other valuable strategies such as validating student expertise and supporting personal designs during class discussion to engage students with computational thinking have been addressed elsewhere (see Fields et al., 2018a, b).

By themselves the strategies discussed above are nothing new in having been identified in much exemplary science and mathematics teaching (see Ball, Thames, & Phelps, 2008). However, it is the application of these teaching strategies to computational thinking that presents a unique and promising approach to develop this emerging field of pedagogy in computer science education. We see them as an example of computational pedagogical content knowledge, or the unique knowledge that teachers need to develop in order to embed computational thinking in their instructional practice to support student learning. This chapter provided salient classroom examples of what teaching iterative practices in the context of personal student projects can look like.

References

Ball, D., Thames, M. H., & Phelps, G. (2008). Content knowledge for teaching: What makes it special? *Journal of Teacher Education, 59*(5), 389–407.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking*. Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada.

Buchholz, B., Shively, K., Peppler, K., & Wohlwend, K. (2014). Hands on, hands off: Gendered access in sewing and electronics practices. *Mind, Culture, and Activity, 21*(4), 1–20.

Buechley, L., Peppler, K., Eisenberg, M., & Kafai, Y. (Eds.). (2013a). *Textile messages: Dispatches from the world of e-textiles and education*. New York, NY: Peter Lang.

Charmaz, K. (2011). Grounded theory methods in social justice research. *The Sage Handbook of Qualitative Research, 4,* 359–380.

CollegeBoard. (2016). *AP computer science principles: Course and exam description effective fall 2016*. CollegeBoard: New York, NY. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58,* 240–249.

Fields, D. A., Kafai, Y. B., Nakajima, T. M., Goode, J., & Margolis, J. (2018a). Putting making into high school computers science classrooms: Promoting equity in teaching and learning with electronic textiles in Exploring Computer Science. *Equity, Excellence, and Education, 51*(1), 21–35.

Fields, D. A., Shaw, M. S., & Kafai, Y. B. (2018b). Personal learning journeys: Reflective portfolios as "objects-to-learn-with" in an e-textiles high school class In V. Dagiene & E. Jastuė (Eds.), *Constructionism 2018: Constructionism, Computational Thinking and Educational Innovation: Conference Proceedings* (pp. 213–223). Vilnius, Lithuania. http://www.constructionism2018.fsf.vu.lt/proceedings.

Fields, D. A., Kafai, Y. B., & Searle, K. A. (2012). Functional aesthetics for learning: Creative tensions in youth e-textiles designs. In J. van Aalst, K. Thompson, M. J. Jacobson, & P. Reimann (Eds.), *The Future of Learning: Proceedings of the 10th International Conference of the Learning Sciences (ICLS 2012), Full Papers* (Vol. 1, pp. 196–203). Sydney, NSW, Australia: International Society of the Learning Sciences.

Fields, D. A., Lui, D., & Kafai, Y. B. (2017). Teaching computational thinking with electronic textiles: High school teachers' contextualizing strategies in *Exploring Computer Science*. In S. C. Kong, J. Sheldon, & R. K. Y. Li (Eds.), *Conference Proceedings of International Conference on Computational Thinking Education 2017* (pp. 67–72). Hong Kong: The Education University of Hong Kong.

Goode, J., & Margolis, J. (2011). Exploring Computer Science: A case study of school reform. *ACM Transactions on Computing Education, 11*(2), 12.

Goode, J., Margolis, J., & Chapman, G. (2014). Curriculum is not enough: The educational theory and research foundation of the Exploring Computer Science professional development model. In *Proceedings of SIGCSE '14* (pp. 493–498). New York, NY: ACM.

Griffin, J., Pirman, T., & Gray, B. (2016). Two teachers, two perspectives on CS principles. In *Proceedings of SIGCSE '16* (pp. 461–466). New York, NY: ACM.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Guzdial, M. (2016). Drumming up support for AP CS principles. *Communications of the ACM, 59*(2), 12–13.

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kafai, Y. B., Fields, D. A., & Searle, K. A. (2014a). Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review, 84*(4), 532–556.

Kafai, Y. B., Lee, E., Searle, K. S., Fields, D. A., Kaplan, E., & Lui, D. (2014b). A crafts-oriented approach to computing in high school. *ACM Transactions of Computing Education, 14*(1), 1–20.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37.

Lui, D., Jayathirtha, G., Fields, D. A., Shaw, M., & Kafai, Y. B. (2018). Design considerations for capturing computational thinking practices in high school students' electronic textile portfolios. In *Proceedings of the International Conference of the Learning Sciences.* London, UK.

Lui, D., Walker, J. T., Hanna, S., Kafai, Y. B., Fields, D. A., & Jayathirtha, G. (in press). Communicating computational concepts and practices within high school students' portfolios of making electronic textiles. *Interactive Learning Environments.*

Margolis, J., & Goode, J. (2016). Ten lessons for CS for all. *ACM Inroads, 7*(4), 58–66.

Margolis, J., Goode, J., & Ryoo, J. (2015). Democratizing computer science knowledge. *Educational Leadership, 72*(4), 48–53.

Mishra, P., & Kohler, M. J. (2006). Technological pedagogical content knowledge: A new framework for teacher knowledge. *Teachers College Record, 108*(6), 1017–1054.

National Research Council. (2011). *Report of a workshop on pedagogical aspects of computational thinking*. Washington, DC: National Academy Press.

Penuel, W. R., Fishman, B. J., Cheng, B., & Sabelli, N. (2011). Organizing research and development at the intersection of learning, implementation, and design. *Educational Researcher, 40*(7), 331–337.

Ragonis, N. (2012). Integrating the teaching of algorithmic patterns into computer science teacher preparation programs. In *Proceedings of ITiCSE '12* (pp. 339–344). New York, NY: ACM.

Soloway, E., & Spohrer, J. C. (Eds.). (1989). *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum.

Tayal, S. P. (2013). Engineering design process. *International Journal of Computer Science and Communication Engineering*, 1–5.

Tofel-Grehl, C., Fields, D. A., Searle, K., Maahs-Fladung, C., Feldon, D., Gu, G., & Sun, V. (2017). Electrifying engagement in middle school science class: Improving student interest through e-textiles. *Journal of Science Education and Technology, 26*(4), 406–417.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49,* 33–35.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16.