Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects

Gayithri Jayathirtha[1*], Deborah A. Fields[2], Yasmin B. Kafai[1]
[1] University of Pennsylvania
[2] Utah State University
gayithri@gse.upenn.edu, deborah.fields@usu.edu, kafai@upenn.edu

ABSTRACT

Debugging, a recurrent practice while programming, can reveal significant information about student learning. Making electronic textile (e-textile) artifacts entails numerous opportunities for students to debug across circuitry, coding, crafting and designing domains. In this study, 69 high school students worked on a series of four different e-textiles projects over eight weeks as a part of their introductory computer science course. We analyzed debugging challenges and resolutions reported by students in their portfolios and interviews and found not only a wide range of computational concepts but also the development of specific computational practices such as being iterative and incremental in students' debugging e-textiles projects. In the discussion, we address the need for more studies to recognize other computational practices such as abstraction and modularization, the potential of hybrid contexts for debugging, and the social aspects of debugging.

KEYWORDS

computer science education, programming, debugging, electronic textiles, making

1. INTRODUCTION

Debugging, the process to fix problems in code that prevent a computer program from functioning as intended, is recognized as a key computational thinking practice in engineering and computing (College Board, 2017; McCauley et al., 2008). In addition to being an important practice, debugging can also illuminate various areas of student struggle and provide opportunities for correction and support (Griffin, 2016). This is evident in studies where novice programmers' errors have illuminated misconceptions about specific concepts such as logical operators or understanding of control-flow statements (e.g. Brown & Altadmri, 2014).

Yet, debugging is an issue not just in computer science but also in engineering education (e.g., Patil & Codner, 2007). Electronic textiles construction kits, that include sewable microcontrollers, sensors, and actuators (Buechley, Peppler, Eisenberg, & Kafai, 2013), bring engineering and computer science together and generate at times interconnected problems for debugging. For instance, during the creation of an e-textile project, problems can occur in the code, in the circuitry, and in the crafting and physical design itself, and students need to test and isolate problems, often fix multiple co-occurring issues that add to the complexity of the project (e.g., Kafai, Fields, & Searle, 2014). Thus these hybrid projects provide an opportunity to promote deeper learning of debugging in engineering and computing, especially if we consider debugging as a type of in-the-moment problem solving of projects (not just code) with errors.

In this paper, we investigate high school students' (14-18 years) debugging in the context of an eight-week long e-textiles curricular unit that took place within three introductory *Exploring Computer Science* classrooms (hereafter *ECS*, Margolis & Goode, 2016). During the unit, students from three classrooms created a series of four open-ended projects of increasing difficulty. In order to understand their debugging more deeply, we studied the problems that students reported they had to debug. Using end-of-unit written portfolios and interviews where students reflected on the challenges they encountered while creating their e-textiles projects, we studied the following questions: What types of challenges did students face, and in what content areas as they were making these projects? What kinds of computational practices did students report in relation to solving problems that came up? What social resources did they draw on to debug projects?

2. BACKGROUND

Debugging has been recognized as a key part of computational thinking for many years. As Papert (1980) noted, "[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). The historical teaching of debugging strategies has focused on helping students discover their own syntax problems (e.g., Robertson et al., 2004) or providing them with strategies for fixing and finding bugs (Carver & Risinger, 1987) through a variety of methods, such as debugging exercises and logs, reflective memos, and collaborative assignments (e.g., Griffin, 2016). Researchers have also developed different technical supports in the form of debugging tools. For instance, Tubaishat (2001) provided tracing tools, while Thomas, Ratcliffe, and Thomasson (2004) offered visualizations and Robertson and colleagues (2004) investigated the timing of interruption tools. Nearly all of this research focused on on-screen programming since it was common in introductory programming courses then. As McCauley and colleagues (2008) noted in their comprehensive review of debugging research, it is unclear how findings and strategies developed from these earlier studies apply to visual programming languages and hybrid construction kits such as e-textiles which also involve collaborative work.

More recently, scholars have started to identify computational *practices* in computer science education, a focus not just on what concepts students are learning but how they are learning it and what thinking strategies they develop. For instance, in their examination of students learning Scratch, Brennan and Resnick (2012) identified four computational practices: being iterative and incremental, testing and debugging, reusing and remixing, and abstracting and modularizing—each of which can result from rich programming experiences. Similarly, Sullivan (2008) outlined seven types of scientific thinking that student exhibited while thinking aloud about solving robotics problems: observing the problem, isolating the problem, generating a hypothesis, testing a hypothesis, controlling variables, manipulating variables, evaluating the solution, and estimating and computing. Together, these studies suggest taking a broader view of the thinking processes that debugging involves.

Several studies have shown that e-textiles can provide a complex context for debugging. The hybrid nature of e-textiles means that problems can occur in several overlapping areas of craft, design, circuitry, and coding (Kafai, Fields, & Searle, 2014; Lee & Fields, 2017). This means that identifying underlying problems is potentially tricky. However, prior studies of debugging in e-textiles have largely focused on areas of circuitry and physical craft, with only elementary computing concepts appearing in studies of debugging (see Litts, Kafai, Searle, & Dieckmeyer, 2016; Fields, Searle, & Kafai, 2016). Lack of time may be a reason for this since most e-textiles projects rarely exceed 16-20 hours of time on projects and rarely include more than one project requiring programming sensors or actuators. In our study, one goal of the e-textiles curricular unit design was to engage students more deeply in computational aspects of e-textiles for more time (roughly 40 hours of class time) with two projects involving coding.

Further, we intentionally looked at whether students discussed getting help from others in their descriptions of debugging in an effort to understand the collaborative nature of debugging. Previous debugging studies have focused mostly on individuals as if learning to debug was solely an individual endeavor (e.g. Fitzgerald et al., 2008). Yet learning in computer science does not happen in isolation. Kafai and Burke (2014) called for a reconceptualization of computational thinking as computational participation, explicitly recognizing the collaborative nature of computing. As collaboration is recognized as a key computational practice for learners to develop (College Board, 2017), some studies have noted the role of others in problem solving with computers or robotics. For instance, Deitrick and colleagues' (2015) analysis of a programming class through a socio-historical lens uncovers the intricacies of collaborative contexts where students, teachers and tools play a definite role in computational learning. Further, Jordan and McDaniel (2014) found that peers serve as a resource for managing uncertainty during problem solving. Yet much more needs to be understood about collaboration with debugging, especially in informal or ill-structured groups (versus pairs or small groups).

3. CONTEXT and PARTICIPANTS

The *ECS* initiative comprises a one-year introductory computer science curriculum with a two-year professional development sequence. This inquiry-based curriculum has been successfully implemented with over 20,000 students. In 2016, we co-developed an e-textiles unit for the *ECS* curriculum and piloted it with two teachers, focusing on teacher practices of making (see Fields, Kafai, Nakajima, Goode, & Margolis, in press). We revised the unit in 2017 and piloted it with three teachers, this time with a focus on student learning (the broader focus of this paper).

The revised unit took place over eight weeks and consisted of a series of four projects: 1) a paper-card using a simple circuit, 2) a wristband with three LEDs in parallel, 3) a classroom-wide mural project where pairs of students created portions that each incorporated two switches to computationally create four lighting patterns, and 4) 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. All the students also documented their projects in portfolios in which they summarized their projects, shared challenges that they faced, and reflected on their learning during the e-textiles unit.

In Spring 2017, three high school teachers, each with 8-12 years of computer science classroom teaching experience, piloted the e-textiles unit in their *ECS* classes in three large public secondary schools in a major city in the western United States. All three 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). In School 1, Angela taught 22 students (6 girls and 16 boys), in School 2, Ben taught 36 students (17 girls and 19 boys), and in School 3, José taught 29 students (20 girls, 9 boys). All names of teachers and students are pseudonyms.

4. DATA COLLECTION and ANALYSIS

Data for this project include all written portfolios submitted by consenting students (69 students from 3 classrooms) and interviews with pairs of students from each classroom (16 students total) discussing problems they encountered while making their e-textiles artifacts. We began analysis by identifying debugging episodes that students reported in their interviews and portfolios. We then grouped these episodes student-wise (69 students), combining two or more challenges whenever a student shared the same issue, both in the interview and the portfolio. This resulted in 210 total debugging episodes.

We coded the debugging episodes in a number of ways, drawing on concepts and frameworks from prior studies whenever applicable. To begin, each episode was classified by content (crafting, circuitry, programming, and design) and then sub-classified within more specific areas of these domains. For instance, we subdivided circuitry based on codes by Peppler and Glosson's (2012) research on e-textiles: connections, polarity, and current flow. For programming, we drew on Brennan and Resnick's (2012) framework of computational concepts: data, events, sequence, conditionals, logic operators, and loops. We also included syntax, an issue specific to text-based programming language. However, with very little prior research done to understand student challenges in designing and crafting, we needed to develop new codes to categorize these challenges, including sewing mechanics, physical construction, and three-dimensional issues of design. Multiple codes could be used for each episode, since areas often overlapped (e.g., a problem involving both circuitry and code). We also included a "general" subcategory in cases of vaguely described problems.

In addition to analyzing content domains, we looked at computational practices students exhibited in their descriptions of the debugging process. For this we used both Brennan and Resnick's (2012) framework of computational practices and Sullivan's further subdivision of problem solving with robotics (see Section 2 for descriptions). Notably, Brennan and Resnick classify "testing and debugging" as one computational practice. However, while problem solving their projects, students often reported practices such as being iterative, so we included all practices identified by Brennan and Resnick and Sullivan in our coding of debugging episodes.

Finally, we considered the larger context of debugging, specifically what resources students used to resolve problems, including digital tools (e.g., Arduino IDE error message bar), physical tools (e.g., seam rippers or curved needles), or social resources (e.g., peers, teachers). Few students reported the use of digital or physical tools. However, many students frequently listed collaboration as a key resource while debugging. Below we share overarching findings from this analysis, focusing on computational concepts, computational practices, and collaborative resources to debug e-textiles projects.

5. FINDINGS

5.1. Computational Concepts Involved in Debugging

In earlier studies of debugging with e-textiles, crafting, circuitry, and simple computational challenges were the primary areas of debugging (Litts et al., 2016; Fields et al., 2016). In this study we found similar reporting of problems that arose in crafting and circuitry, but we also identified two other areas of debugging that were not discussed in earlier studies. First, students in our study reported *coding* challenges almost as often as crafting or circuitry and this highlighted some key coding concepts. Second, students also encountered new challenges in *three-dimensional design*. We describe these two areas in more detail below.

Among the 210 total debugging episodes, concepts discussed were almost evenly distributed across coding (29%), crafting (30%), and circuitry (28%). Within the episodes that discussed coding challenges and resolutions, a wide variety of concepts were reported, ranging from simple problems with syntax and labeling to more advanced issues with logical operators and control-flow statements. Forty-three students across three classes mentioned coding challenges at least once: a total of 61 episodes. Of these debugging episodes focused on code, 64% of included "simple" issues that involved syntax, mislabeling variables or incorrect usage of constants. For example, some of these bugs included fixing brackets in conditional statements and functions, and mislabeling a sensor as "OUTPUT" instead of "INPUT." While these are still relatively simple issues, resolving syntactical and labeling bugs such as these is a key practice in coding (McCauley et al., 2008).

However, 36% of the coding issues shared revolved around more complex computational concepts such as determining mathematical expressions for ranges of sensor values and managing multiple conditional statements. Consider David (School 1), who had difficulty determining the most effective ranges for his human sensor project. This project included two conductive patches that created a range of numerical values depending on how hard someone squeezed. Students had to create four ranges of these values and program them to trigger different lighting patterns. As David expressed, "it was harder to think of how big your range had to be so that it would actually react to how you want it to be." After he realized his first attempt at coding ranges was inadequate, he iteratively tested the sensor, and represented a sequence of readings on a number line. Many students struggled with coding the ranges on their patches and took substantial time to fix them. Other more complex challenges that students faced included organizing multiple conditionals, especially if they involved two stages (i.e., using "if___, else___" instead of just a series of "if" statements), using additional sensors (e.g., light sensor) or in-built functions (e.g., random number generator). The variety and relative complexity of coding challenges reported by students highlight the affordances of e-textiles to support debugging both simple and advanced computational coding concepts.

Besides struggles with coding, another new area of struggle involved designing circuits on a three-dimensional artifact such as a stuffed animal or sweatshirt, especially common in the human sensor project. These designs required students to plan their circuitry two-dimensionally on paper but translate it onto a three-dimensional item. This posed new challenges to students. Thirteen of the 69 students (19%) specifically mentioned this issue within their debugging. For instance, while making his "Angry Bird" stuffed animal project, Rodrigo (School 1) realized he had to change his circuitry once he started working in three dimensions. "I made these changes because it was difficult planning out a 3D model on paper and if I hadn't made changes to the pin numbers, then the paths would have crossed," he explained. Photos from his portfolios are visible in Figure 1, where he showed two sides of the stuffed animal as well as his final circuitry diagram highlighting those same two sides (front and bottom). Though issues of three-dimensional circuitry design have not appeared previously in work on learning with e-textiles in K-12 education, it has come up with university students during clinical interviews (Lee & Fields, 2017), suggesting it may be an area of debugging that students face while working on more advanced projects. This also raises opportunities to consider spatial thinking in e-textiles design.

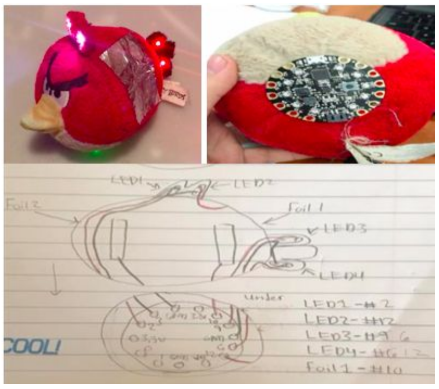

Figure 1. Rodrigo's Angry Bird project (top left to right clockwise): Upper view; bottom view (showing microcontroller); Circuit diagram.

5.2. *Computational Practices Related to Debugging*

In addition to content areas of debugging, we also sought to better understand the process of debugging, analyzing this through the computational practices lenses. Out of 69 students, 60 shared at least one of the four standard computational practices suggested by Brennan and Resnick (2012) in their framework. Out of these four practices, testing and debugging was the most mentioned (47 students), followed by iterative and incremental practices (35 students). The two other practices, abstraction and modularization, and reusing and remixing were rarely discussed. This may be because of how the questions were phrased in interviews and in the portfolio, which focused on challenges students faced. For instance, in their focus on problems, students did not mention remixing designs although remixing and reusing daily-use items such as backpacks and soft toys was an integral part of their human sensor project. Further, though there were opportunities for applying abstraction and modularity (i.e., breaking down a project and/or code into parts), this did not seem to be a conscious way that students thought about this process with regard to problem solving. However, yet another computational practice that emerged from student descriptions was collaboration, which is also presented as a perspective in Brennan and Resnick's (2012) framework. Thirty-six students reported on collaboration as an integral aspect of fixing errors, leading us to suggest collaboration as more of a computational practice rather than a perspective developed, which we will elaborate shortly.

Though all debugging episodes concerned students fixing issues, in some instances students shared more specific details about how they identified, isolated, and otherwise focused on understanding a particular problem. In these 47 instances, we coded for specific areas that Sullivan (2008) identified. The most prominent of these were observing the problem (46 students), isolating the problem (43 students), and generating a hypothesis about the cause of the problem (35 students). As an example, consider how Alexa and Antonio (School 2) worked through a series of circuitry problems in their Pacman-themed mural project (see Figure 2). As Alexa expressed in her portfolio: "[In] our first design we wanted the playground on the back of project. When we tried that, the conductive thread crossed each other… We dealt with our problem by redesigning our project, so that the playground was in the front and the conductive thread wasn't touching." Alexa and Antonio first observed the source of the error as the short-circuit (crossed threads) and hypothesized that the spatial placement of the Circuit Playground (microcontroller) at the back of their Pacman mat was causing the short circuit. They were able to isolate specific locations where these short circuits occurred and plan their next iteration to fix them.

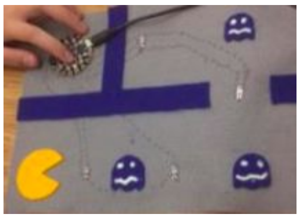

Figure 2. Alexa and Antonio's Pacman project

Along with testing and debugging, being incremental and iterative was another other key computational practice evident in student narrations. Of the 35 students who shared about this, 29 discussed incrementally revising their project design and 10 shared about repeatedly testing their sensor values and adjusting their project code to suit the varying values. (Note: we classified *repeated* testing of a problem under iteration rather than testing and debugging). One of the key challenges underlying revisions was translating project plans from paper representations to physical artifacts. As previously mentioned, many students had to revise their project upon realizing that their plan on paper did not work when sewn in three dimensions. For instance Alma (School 2) expressed that "[W]hen sewing [our project] we realized that everything was basically backwards" and had to substantially change the placement of each LED so to have "clean lines" without short circuits.

Besides design translations, the other major area of being iterative and incremental was in testing the sensor patches. Here David (School 1) again provides an explanation for what iterative testing looked like:

> So from my last project, it was a human sensor and my scales were… pretty much wrong to the point where only one pattern worked… [T]o fix the problem… I slowly started testing out. So, I touched it. Okay, this is the values for a light touch, just inputted that. I said, 'let's squeezed it harder.' [sic] I looked at the values, and inputted that… As I looked at the values, I am like, okay, the range from this to the next pattern, it's kinda too small. So I have to make it bigger so that it can be a bit more sensitive.

This encouraging example of iteration demonstrates the careful way that some students had to work to program their sensors. Often their first attempt would result in poorly thought-out ranges, and, like David, students had to proceed through cycles of testing and adjusting the range of values corresponding to squeezing. Though only 10 students described this particular process, it is a practice that could be expanded on more intentionally in future iterations of the curriculum and in debugging pedagogy more generally.

5.3. Collaboration Contexts Related to Debugging

One unexpected finding was how often students' debugging involved collaboration with classmates, partners and teachers. Most students (75%) explicitly mentioned help they received from peers or a teacher in at least one of the challenges they described (in 36% of the challenges overall). Unlike an earlier study that observed low peer collaboration in e-textiles (Litts et al., 2016), this analysis revealed student engagement with different types of collaborators throughout their e-textiles debugging, from their immediate partners on a project, to students at the same table, to the wider class community.

Students reported different kinds of supports that they received from peers and teachers across a range of issues—from identification of syntactical errors to understanding concepts such as conditional statements. An example for a simple support includes Ethan's (School 3) reporting of dim lights in his quilt project. His classmate helped him locate and isolate the problem: missing a line in the setup section of the code that initialized the pin to OUTPUT. Students also mentioned getting help with more complex struggles. For instance, Allie (School 2) used her classmates to test the sensors of her human sensor project, using "different people's pressure" and changing the ranges in her project. Surprisingly, students rarely mentioned teacher participation in debugging (close to 11% of challenges).

Collaboration was mentioned frequently in students' reports of debugging although students were graded individually for this unit. That so much collaboration was evident in these contexts suggests that there is much more to discover about unstructured peer-to-peer debugging in students' e-textiles design processes and in debugging open-ended computational projects.

6. DISCUSSION

Our analysis of student challenges and solutions demonstrates that debugging open-ended e-textiles projects can provide a rich context for students to experience a range of computational concepts and practices. Our study noted promising new areas of conceptual struggles for e-textiles students, specifically in the domains of coding and three-dimensional design. We think this is because students were able to go deeper in these areas with two advanced e-textiles projects compared to prior studies that only had one such project (e.g., Fields et al., 2016; Litts et al., 2016). This suggests that pursuing a *series* of challenging e-textiles projects may provide more opportunities for deeper learning of computing concepts and practices than just one or two projects. It also raises the potential for supporting debugging more generally by creating a series of projects in other computational domains, not just e-textiles.

In addition to conceptual learning, students in this study reported using certain computational practices such as being iterative, testing and debugging, and collaboratively problem solving. Interestingly, within the area of debugging, students' reports consistently highlighted the need to identify and isolate problems, something that should not be trivialized. Unlike other studies of debugging that focus solely on debugging code (e.g., Brown & Altadmri, 2014), students with e-textiles projects had to consider the origin of a bug from among several possibilities: code, circuitry, craft, or spatial design. Yet, we also recognize that this study was limited to students' *reporting* of bugs rather than a study of observing of how they actually solved them. This opens up the need for deeper research on students' in-the-moment debugging to see whether students engage in other steps of debugging such as manipulation of variables, evaluation of solutions, and estimation of data.

One other key finding was frequent student collaboration during problem solving. Students shared collaboration not only at the level of formal pairs and small groups but within the broader classroom, turning the class into a community of learners. The physical layout of the classroom with tables and shared supplies along with the teachers' allowing students to move between tables may have encouraged this fluid collaboration (Fields et al., in press). More so, these findings call for a reconceptualization of collaboration in these spaces to better understand the roles taken on by different participants. A closer look at these types of settings may help us understand and classify different kinds of supports students provide to each other. Such an analysis could also help us understand the supportive role of teachers in creating collaborative classrooms, informing the development of new pedagogical approaches for students and professional development for teachers.

The interdisciplinary nature of e-textiles provided a unique opportunity to study debugging in a hybrid context. If debugging is a core area of computation, then as a field we need to look beyond code-only settings of computation to hybrid settings (including but not limited to e-textiles) where students are introduced to debugging in more challenging situations which demand multiple iterations of revising and testing. Further, more studies of debugging are needed in many contexts that look at it less as an individualistic and more as a social practice, moving from computational thinking to computational participation (Kafai & Burke, 2014).

7. ACKNOWLEDGEMENTS

This work was supported by grants from the National Science Foundation to Yasmin Kafai, Jane Margolis, and Joanna Goode (# 1509245), and Yasmin Kafai and Mike Eisenberg (#1742140). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF, the University of Pennsylvania, or Utah State University. Special thanks to Tomoko Nakajima for her help with data collection and to Debora Lui, Justice T. Walker, and Mia Shaw for their valuable feedback.

8. REFERENCES

Brennan, K. and 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.

Brown, N. C., & Altadmri, A. (2014, July). Investigating novice programming mistakes: Educator beliefs vs. student data. In *Proceedings of the tenth annual conference on International computing education research* (pp. 43-50). New York, NY: ACM.

Buechley, L., Peppler, K. A., Eisenberg, M. & Kafai, Y. B. (Eds.) (2013). *Textile Messages: Dispatches from the Word of Electronic Textiles and Education*. New York, NY: Peter Lang Publishers.

Carver, S. & Risinger, S. (1987). Improving children's debugging skills. In G. Olson, S. Sheppard & E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop* (pp. 147-171). Norwood, NJ: Ablex.

College Board (2017). *Advanced Placement Computer Science Principles Course Guide*. Retrieved from https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf

Deitrick, E., Shapiro, R. B., Ahrens, M. P., Fiebrink, R., Lehrman, P. D., & Farooq, S. (2015, July). Using distributed cognition theory to analyze collaborative computer science learning. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 51-60). New York, NY: ACM.

Fields, D. A., Searle, K. A., & Kafai, Y. B (2016). Deconstruction kits for learning: Students' collaborative debugging of electronic textile designs. In *FabLearn '16, Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education* (pp. 82-85). New York, NY: ACM.

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

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93-116.

Griffin, J. M. (2016, September). Learning by taking apart: deconstructing code by reading, tracing, and debugging. In *Proceedings of the 17th Annual Conference on Information Technology Education* (pp. 148-153). New York, NY: ACM.

Jordan, M. E., & McDaniel Jr, R. R. (2014). Managing uncertainty during collaborative problem solving in elementary school teams: The role of peer influence in robotics engineering activity. *Journal of the Learning Sciences*, *23*(4), 490-536.

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

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

Lee, V. R. & Fields, D. A. (2017). Changes in undergraduate student competences in the areas of circuitry, crafting, and computation after a course using e-textiles. *International Journal of Information and Learning Technology*, *34*(5), 372-384.

Litts, B. K., Kafai, Y. B., Searle, K. A., & Dieckmeyer, E. (2016). Perceptions of productive failure in design projects: High school students' challenges in making electronic textiles. *International Conference of the Learning Sciences*, 498-505.

Litts, B. K., Kafai, Y.B., Lui, D. A., Walker, J. T., & Widman, S.A. (2017). Stitching codeable circuits: high school students' learning about circuitry and coding with electronic textiles. *Journal of Science Education and Technology*, *26*(5), 494-507.

Margolis, J., & Goode, J. (2016). Ten Lessons for CS for All. *ACM Inroads*, *7*(4), 58-66.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67-92.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Patil, A., & Codner, G. (2007). Accreditation of engineering education: review, observations and proposal for global accreditation. *European Journal of Engineering Education*, *32*(6), 639-651.

Peppler, K., & Glosson, D. (2012). Stitching circuits: Learning about circuitry through e-textile materials. *Journal of Science Education and Technology*, *22*(5), 751-763.

Robertson, T., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, F., Beckwith, L., et al., (2004). Impact of interruption style on end-user debugging. In E. Dykstra-Erickson & M. Tscheligi (Eds.). P*roceedings of CHI'04* (pp. 287-294). New York, NY: ACM.

Sullivan, F. R. (2008). Robotics and science literacy: Thinking skills, science process skills and systems understanding. *Journal of Research in Science Teaching*, *45*(3), 373-394.

Thomas, L., Ratcliffe, M. & Thomasson, B. (2004). Scaffolding with object diagrams in first year programming classes: Some unexpected results. *ACM Inroads, 36*(1), 250-254.

Tubaishat, A. (2001). A knowledge base for program debugging. In *Proceedings of the International Conference on Computer Systems and Applications* (pp. 321-327). Beirut: IEEE Press.