# Learning by Fixing and Designing Problems: A Reconstruction Kit for Debugging E-Textiles

Full Paper

**Debora Lui**
University of Pennsylvania
3700 Walnut St.
Philadelphia PA, USA
deblui@upenn.edu

**Emma Anderson**
University of Pennsylvania
3700 Walnut St.
Philadelphia PA, USA
emmaa@upenn.edu

**Yasmin B. Kafai**
University of Pennsylvania
3700 Walnut St.
Philadelphia PA, USA
kafai@upenn.edu

**Gayithri Jayathirtha**
University of Pennsylvania
3700 Walnut St.
Philadelphia PA, USA
gayithri@gse.upenn.edu

## ABSTRACT

In this paper, we present the development of a "reconstruction kit" for e-textiles, which transforms fixed-state construction kits—maker tools and technologies that focus on the creation of semi-permanent projects—into flex-state construction kits that allow for endless deconstruction and reconstruction. The kit uses modular pieces that allow students to both solve and create troubleshooting and debugging challenges, which we call "DebugIts." We tested our prototype in an after-school workshop with ten high school students, and report on how they interacted with the kit, as well as what they learned through the DebugIt activities. In the discussion, we delve into the affordances and challenges of using these kits as both learning and assessment tools. We also discuss how our pilot and prototype can inform the design of reconstruction kits in other areas of making.[1]

## CCS CONCEPTS

• **Social and professional topics** → **Computer Science education**; Computational thinking; *K-12 Education*

## KEYWORDS

Debugging, Computational Thinking, Making, Productive Failure

**ACM Reference format:**
Debora Lui, Emma Anderson, Yasmin B. Kafai, Gayithri Jayathirtha. 2017. Learning by Fixing and Designing Problems: A Reconstruction Kit for Debugging E-Textiles. In *Proceedings of FabLearn17, October 21-22, 2017.* 8 pages.
https://doi.org/10.1145/3141798.3141805



*FabLearn '17, October 21–22, 2017, Stanford, CA, USA*

*ACM ISBN 978-1-4503-6349-5/17/10…$15.00*
*https://doi.org/10.1145/3141798.3141805*

## 1 INTRODUCTION

Making has become popular within educational sites because of its ability to promote science, technology, engineering, and math (STEM) learning and computational thinking (CT) skills [1, 10]. Many efforts to support maker activities have focused on the design of "construction kits," which Resnick and Silverman [20] describe as "systems that engage kids in designing and creating things" (p. 1). These kits can be situated on-screen (e.g., Scratch, StarLogo), off-screen (e.g., Lego Bricks, littleBits), or in combined 'hybrid' spaces with software and hardware elements (e.g., Lego Mindstorms, Arduino). Designing high quality construction kits requires creating easy-to-use interfaces and materials that allow even novice makers to design and create a wide range of projects to learn about powerful ideas in STEM disciplines such as feedback or complex systems [16].

While there is much emphasis on construction in the Maker Movement, we argue that equal attention should be paid to the intermediate processes of making, such as dealing with unexpected problems that inevitably arise along the way. Of particular relevance here are the practices of troubleshooting, debugging and problem solving at large. As Papert [16] observed early on "when you learn to program a computer, you almost never get it right the first time (p. 23). Thus, troubleshooting always involves taking something apart to some extent—whether a program or a seam—and putting it together again once one has figured out the issue. In other words, making is just as much about construction as it is about deconstruction and therefore, *re*construction.

Some construction kits, which can be called *flex-state* construction kits, are specially designed to allow makers to continually explore within this space of deconstruction and reconstruction, and consequently troubleshooting and

debugging. For instance, children using Legos are encouraged not only to build creations, but also break and remake them. Other construction kits, which can be labeled *fixed-state* including Adafruit FLORA and MakerBot, focus more on the creation of seemingly permanent artifacts, whether light-up hoodies or architectural models. Due to this attention on final rather than ephemeral products, fixed-state kits limit the opportunities for endless reconstruction as seen with flex-state kits. Furthermore, the process of debugging within fixed-state construction kits is often arduous; fixing a 3-D print, for example, involves going back to a program file, figuring out if the error is in the file or the printer, addressing, the issue, and then printing the object all over again. While the creation of personal artifacts is a hallmark of the Maker Movement, we argue that this lack of emphasis on deconstruction and reconstruction misses a rich opportunity for learning and assessment.

In this paper, we aim to include deconstruction, reconstruction, troubleshooting and debugging within the whole cycle of making by proposing what we call a *re*construction kit. By adding modular moveable elements, a reconstruction kit can transform as fixed-state kit into a flex-state one. Here, we describe the design and testing of a first prototype of an electronic textiles (e-textiles) reconstruction kit, based on LilyPad Arduino [3]. An extension of the Arduino microcontroller, the LilyPad allows makers to create fabric-based electronics projects using sewable components such as LEDs, buzzers, and switches. Through creation of e-textiles projects, students learn to integrate multiple domains of knowledge and skill including design, circuitry, coding, and crafting [12]. We define the LilyPad Arduino as a fixed-state construction kit because it requires sewing things together in order to create semi-permanent connections. Debugging within this space is often tedious and time-consuming since it involves taking out and re-doing stitches from a sewn circuit. Our kit bypasses this problem by turning e-textiles components into modular pieces, thus allowing for flexible de/reconstruction. We report here on a workshop in which seven teams of high school students used this e-textiles reconstruction kit for the purposes of learning through debugging. Using the kit, we developed a series of challenges (or "DebugIts" [2]) each focused on a particular issue in circuitry or coding. After students solved these, we then asked them to construct their own DebugIt challenges for others to tackle. We build on a previous study exploring e-textiles debugging activities [8] to ascertain how our reconstruction kit works as a viable tool to both teach and assess student knowledge of problem solving.

## 2 BACKGROUND

Debugging has long been considered an important skill to support within computer science learning (e.g., [4, 17]). Researchers have developed a range of tools and methods to support this on-screen skill, for example, development logs, reflective memos, tracing tools, and visualizations (e.g., [2, 5, 9]). However, as McCauley and colleagues noted, it is unclear how the findings and strategies developed from these studies apply to different computational contexts, such as ones that encompass both on and off-screen elements [15]. In focusing on these 'hybrid' designs, we posit there is potential to promote deeper problem-solving skills through the process of debugging.

Of particular note here is students abilities to develop their computational thinking (CT) skills, a problem-solving approach that has recently gained traction within educational contexts [11]. As defined by Wing, CT moves beyond knowing the specifics of code or programming to an entirely different way of approaching problems [24]. This can include, for instance, thinking about the component parts of a system and how they fit together to form a complete solution. These aspects become particularly relevant in the context of e-textiles, where one must consider the interface between the *on-screen* world of code, and the *off-screen* world of circuitry and crafting in order to create a functioning computational artifact [7, 12].

Moreover, the process of debugging also encompasses a mode of problem solving that Kapur calls "productive failure" [13]. This concept describes the counterintuitive notion that students can potentially learn more by moving through a series of struggles and failures rather than being carefully scaffolded through incremental, correct steps. While Kapur focused on this idea within the context of ill-defined problems, others have spoken about how productive failure has just as much potential within the arena of open-ended design activities [14]. For instance, this becomes particularly apparent in maker contexts where creators often have to deal with a range of different, often finicky materials that require individuals to tinker, troubleshoot, and fail before creating a working project [18, 19].

These considerations of debugging as a form of computational thinking and productive failure thus inspired the development of a reconstruction tool for e-textiles learning. We build off prior work where we implemented a debugging challenge using pre-sewn, pre-programmed e-textile projects that contained a curated collection of circuit and code problems [8]. For circuitry, this included short circuits, electric topology, and polarity. For coding, this included constant versus variable pins, control flow, and end-state definition. Students developed particular methods of solving these challenges, including strategically isolating and prioritizing issues, and running through cycles of hypothesis making and testing. While students managed to solve many of these problems, the static nature of the original tool did not allow students to tinker or experiment with their solutions, a key practice of making and debugging. Additionally, the process was not particularly creative; while flex-state construction kits are often focused on "design for designers" [20], or avenues for creative expression, in these debugging challenges, students were mostly expected to provide the single correct answer rather than developing alternate solutions or even challenges of their own.

[2] The name DebugIt makes reference to the Debug-It Studio of buggy Scratch projects that Brennan released on the Scratch Ed website beginning in 2010 so that other Scratch members could solve them [6].

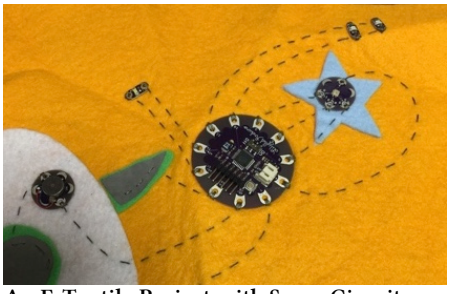

Figure 1: **An E-Textile Project with Sewn Circuits**

Figure 2: **Reconstruction Kit for E-Textiles. Functional LilyPad circuit (top); modular felt strips with metallic thread (bottom left) and hooked LEDs (bottom right) to bypass sewing**

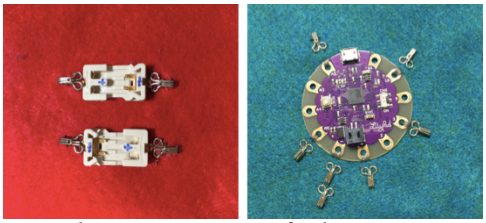

Figure 3: **Simple construction mat for battery-powered circuits (left); complex construction mat with LilyPad Arduino for codeable circuits (right)**

Our reconstruction kit is designed to address these elements by transforming previously fixed pieces into modular pieces. Here, stitched circuits with metallic thread become felt strips and hooked LEDs that can easily be attached or detached (Figure 2), therefore eliminating the time-consuming need to sew (or unsew). Using the kit, students can not only build functional e-textiles circuits rapidly, but they can also encounter, solve, and create a wide range of debugging issues—something we piloted through our workshop. In this study, we therefore address the following research questions: 1) How did students solve and create the DebugIt challenges using the reconstruction kit? 2) What were student perceptions regarding the debugging activities and reconstruction kit in terms of how these helped or hindered their learning?

## 3 THE DEBUGIT RECONSTRUCTION KIT

Our reconstruction kit is based off the LilyPad Arduino construction kit: a series of sewable electronics including an Arduino-based microcontroller, sensors (e.g., light, temperature) and actuators (e.g., LEDs, buzzers). In its regular usage, one sews components together in circuits using conductive thread into fabric (Figure 1). To bypass the complexity of sewing, our kit uses hooked LEDs and buzzers along with felt strips and conductive thread to allow for quick connections and disconnections (Figure 2). The kit also includes two construction mats on which pieces can be connected: a simple construction mat with two coin cell battery holders, and a complex construction mat with a LilyPad Arduino microcontroller (see Figure 3). With the simple mat, users can create simple circuits with LEDs and buzzers powered by 3V coin cell batteries. With the complex mat, users can additionally create codeable circuits controlled by Arduino programs. Building off our previous work [8], the kit can also support the creation of debugging challenges, which we call "DebugIts," where students can work on fixing mistakes within e-textiles circuit or code (Figure 4).

In creating our kit, we followed principles that Resnick and Silverman have outlined for successfully designing construction kits [20]. This includes thinking carefully about what technologies/skills we chose to "black box" (i.e., sewing), providing "low floors" for easy access (hooked rather than sewn connections), along with "wide walls" for variety and exploration (construction mats as the basis for multiple projects), and finally, an emphasis on "design for designers" (re-usability of the pieces allowing for endless redesigns and reconstructions).

## 4 METHODS

### 4.1 Participants and Workshop Design

We conducted an after-school workshop at a science museum in a northeastern U.S. city with ten high school freshmen from a science magnet public school. Based on surveys, five students had prior experience with circuitry (mostly through school), and six had prior experience with coding (mostly through the "Hour of Code" program by Code.org). One student had minimal prior experience with e-textiles (sewing a bracelet), but the rest were working with e-textiles for the first time.

The workshop met four times (once a week for 110 minutes) and was led by three authors of this paper. In small groups of one to three, students engaged with both the simple and complex construction mats. Groups were led through a sequence of four activities per mat: (1) creating a simple or codeable e-textiles circuit from scratch, (2) solving an instructor-designed Debug-It challenge, (3) designing their own DebugIt challenge, and (4) solving a learner-designed DebugIt challenge. For this paper, we focus on the last week of the workshop, where students solved and created DebugIts using the complex mat. In prior weeks, students had learned how to create working e-textiles circuits

and write and modify code. During the last session, students started off working in seven groups; however, in between challenge A2 and B1, two groups merged due to laptop issues.

After solving these challenges, students were asked to create their own DebugIt challenges. In order to help guide students, they were first asked to create a working circuit and code combination, and then asked to add a 'bug' to either the code or the circuit. They were also asked to designate a final state (e.g., all three lights should turn on). Groups then traded DebugIts with others.

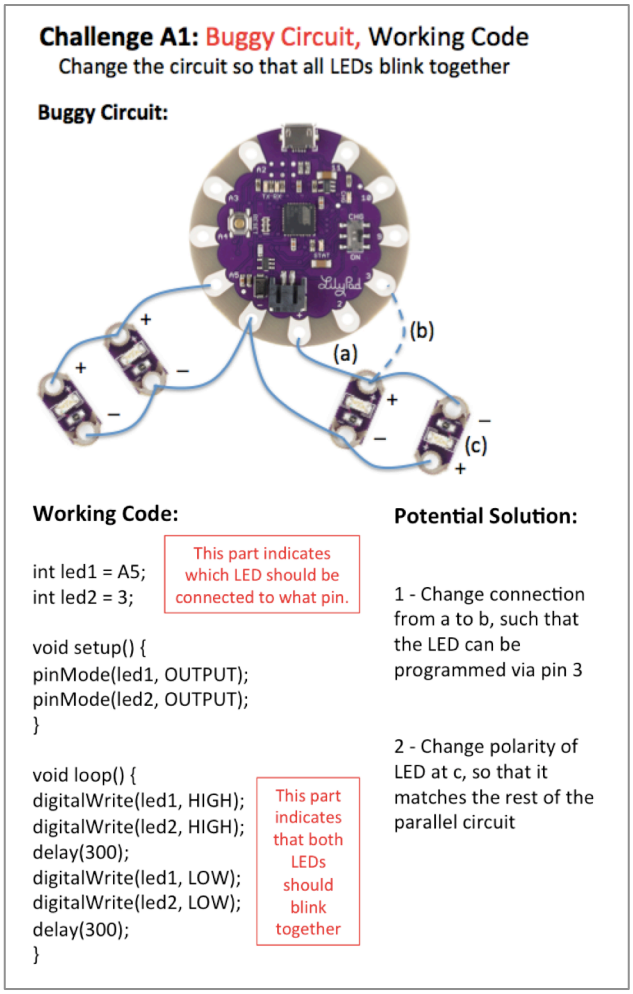


**Figure 4: An Instructor-Designed DebugIt including a 'buggy' circuit that contains incorrect connections based on the accompanying code. A potential solutions is included here.**

## 4.2 DebugIt Design

For the instructor-designed DebugIt challenges, students were provided either a circuit or code that had mistakes (or 'bugs') and asked to fix these with the least number of 'moves' (defined as unhooking and hooking a piece, or changing a line of code). Students solved the challenge once they reached a predesignated final state (e.g., all four lights blink together) (see Figure 4 for a sample DebugIt). While the simple mat DebugIts only involved circuit-based bugs, the complex mat DebugIts involved fixing either the code or the circuit. Each challenge given to the students focused on a particular skill and/or area of understanding of e-textiles (see Table 1). Challenges increased in difficulty each day from the simplest to the most complex: for instance, while A1 deals with parallel circuits that are programmed together, B1 deals with independent circuits that can be programmed separately, something that requires greater knowledge of both circuitry and code.

**Table 1: Instructor-Designed Complex Mat DebugIts and Issues**

| Name | Description | Type of Issue Addressed |
|---|---|---|
| A1 | Buggy Circuit, Working Code | Polarization with parallel circuit: changing the connection points of an LED so that its polarity matched the rest of the parallel circuit; Electronic topology with parallel circuits: changing a connection from a constant (+) to a variable pin, so that a parallel led circuit could be programmed together. |
| A2 | Working Circuit, Buggy Code | Variable declaration/interfacing: changing a variable so that it matched with the given circuit; End state definition within a function: changing a function so that the circuit would result in the desired final state (e.g., all LEDs on). |
| B1 | Buggy Circuit, Working Code | Polarization with independent circuit: changing the connection points of an LED so that its polarity matched another independent circuit (grounded together); Electronic topology with independent circuits: changing a connection so that LEDs in independent circuits could be programmed separately. |
| B2 | Working Circuit, Buggy Code | Setting up program constants: setting a pin as an electrical output; End state definition within the sequence" changing a sequence of code so that the circuit would result in the desired final state (e.g., 4 LEDs blinking together). |

## 4.3 Data and Analysis

We draw from two sources of data to answer our research questions: student worksheets and a survey. For the last day of the workshop, there were six different worksheets per group: one for each of the instructor-designed DebugIts (4), one focused on designing a DebugIt challenge, and one focused on solving other students' designed DebugIt (see Figures 5 and 6). For all the DebugIts solved (whether instructor or learned-designed), the worksheets asked students to report on their corrections to either the circuit or the code, and a self-reported number of moves. We recorded how many groups attempted and solved each instructor DebugIt and looked at the range and average of their self-reported number of moves. We also examined how their solutions deviated from our expected solution, both quantitatively (looking at the number of moves made) and

qualitatively (type of moves). In the worksheet, we also asked students to write what they thought was the most difficult aspect of each DebugIt. Based on their answers, we looked for emerging themes to help understand the affordances and challenges of the kit.

For the DebugIts that groups designed themselves, the worksheets asked for their buggy code or buggy circuit along with the corrected version. We counted how many groups created buggy circuits versus buggy code, and examined the bugs themselves, looking at what kinds of issues groups decided to highlight (e.g., end state definition within the function, polarization), in order to compare them with the instructor-created DebugIt challenges. We counted how many of these were solved, and recorded the number of self-reported moves. We also looked through their answers regarding the "most difficult" part about the challenge in order to highlight emerging themes.

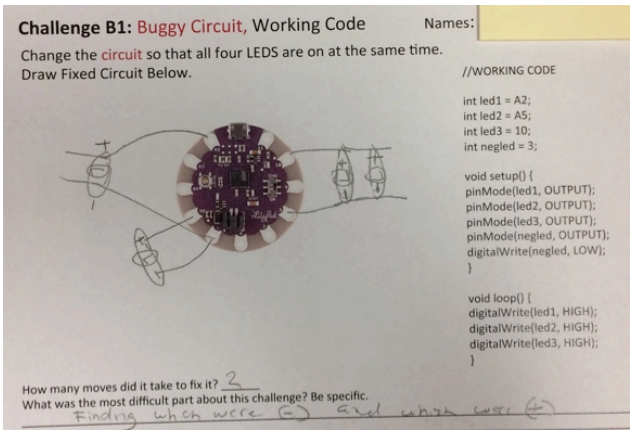


**Figure 5: Student worksheet showing a solution for Challenge B1, with drawing of the corrected circuit, number of moves made, and discussion of the most difficult part of the challenge.**

**Figure 6: Students working together to fix a circuit on the complex reconstruction mat.**

To capture student perceptions of the activities, we administered an exit survey that asked which activity was most and least helpful to their learning and why. This included: creating a codeable circuit, solving a buggy code, solving a buggy circuit, or creating a DebugIt. All ten students answered these questions. We tallied up responses received for each activity, and identified emerging themes across their explanations. We also asked them what they felt they learned about coding and circuitry within the workshop. Only eight of ten students answered these questions due to time constraints.

# 5 FINDINGS

## 5.1. Fixing Instructor-Designed DebugIts

Student groups were allotted up to 60 minutes to work on the instructor DebugIts (A1, A2, B1 and B2). Within this given timeframe, all groups were able to solve A1, A2 and B1, while only one group attempted, but did not solve B2 (see Table 2).

Looking at the number of moves, we expected students to use two to three moves per challenge. However, groups' self-reported number of moves ranged from one to ten, with the most number of moves reported for Challenge B1 (see Table 2). This finding makes sense given that B1 was the most complex challenge the groups encountered, and was the site where they tinkered and experimented the most with the moves they took (as opposed to earlier challenges, where they felt more confident).

**Table 2: Groups who attempted/solved DebugIts**

| DebugIt | Groups Attempted | Groups Solved | Alternate Solutions | Moves Made |
|---|---|---|---|---|
| **A1** – Buggy circuit | 7 | 7 | 2 | Range: 1-4<br>Average: 2.57<br>Expected: 3 |
| **A2** – Buggy code | 7 | 7 | 1 | Range: 1-4<br>Average: 2<br>Expected: 2 |
| **B1** – Buggy circuit | 6 | 6 | 2 | Range: 1-10<br>Average: 4.33<br>Expected: 3 |
| **B2** – Buggy code | 1 | 0 | N/A | N/A |

Looking at their solutions themselves, we also calculated the number of "alternate" solutions provided per DebugIt, meaning ones that deviated from what we had originally intended. For Challenge A1, the intended final solution included two parallel circuits; however, two groups provided only one parallel circuit, which still fulfilled the intended final state (all LEDs blinking together). For Challenge A2, only one group provided an alternate solution; while they added seemingly redundant lines of code, it still led to the correct final state. For Challenge B1, two alternate solutions were provided. One involved creating an 'always-on' rather than codeable connection, and the other involved an entirely different circuit diagram. Again, the existence of these alternate solutions seem to highlight students' freedom to experiment and tinker while solving these DebugIts, often providing creative solutions beyond what was expected.

Looking at reports of their difficulties with DebugIts, students mentioned a few issues, including not understanding polarity within circuits, and being "worried about making too many moves." Several groups mentioned dealing with the physical challenges of the kit itself, specifically the thickness of the felt conductive strips, and the tenuous nature of the hooked

connections between components. However, the most common difficulty described by groups was understanding the relationship between the given code and the circuit. For example, one group wrote the most difficult part of the buggy code challenge was, "understanding what everything meant," or specifically reading parts of the code to "see what had power" or "what's active" on the LilyPad. For another group, this became clear through their process of solving the buggy circuit of A1. At first, the group created an incorrect circuit that did not solve the challenge. However, after reading and interpreting the code they were able to provide a revised and correct solution to the challenge. As the group noted: "[We] didn't see this at first" with an arrow pointing to a line of the program, thus illustrating their realization that the code is intricately connected to the function of an e-textiles circuit. Working with the DebugIts thus forced students to deal with the interface between on and off-screen elements, a key component of computational thinking.

**Table 3: Description of Learned-Designed DebugIts**

| Group | Challenge Created | Type of Issue | Solved? | Moves Made |
|---|---|---|---|---|
| 1 | Buggy circuit | Electronic topology with parallel circuit | Yes | 2 |
| 2 | Buggy code | End-state definition within the sequence (or incomplete code) | No | N/A |
| 3 | Buggy code | End-state definition within the function; variable declaration/ interfacing; setting up program constants | Yes | 6 |
| 4 | Buggy code | End state definition within the function | Yes | 3 |

## 5.2 Creating/ Fixing Learner-Designed DebugIts

For the learner-designed DebugIts, only four of six groups were able to complete the task of creating and trading challenges within the allotted 30 minutes. Of the groups that finished, one group chose to create a buggy circuit-working code combination, while three decided upon a working circuit-buggy code combination (see Table 3). Looking at the bugs themselves, two of the four groups (1 and 4) presented issues that replicated the instructor-designed DebugIts solved earlier (e.g., electronic topology with parallel circuit, end state definition within the function). Group 2, however, created an unconventional challenge that involved adding lines to already working code for a new end state (to make the LEDs blink, rather than staying on). While this is not necessarily a traditional 'bug' or mistake, 'solving' this DebugIt did require a more complex understanding of programming, something not completed by the receiving group. Group 3, by far, had the most unique challenge (see Figure 7) since their code and circuitry did not follow the standards presented in earlier challenges: code variables were redefined within the loop despite being defined earlier, the circuit contained a crossed wire (which actually worked because of the insulation provided by the felt strip), and contained a seemingly redundant ground pin. Despite these deviations from convention, their DebugIt still worked and was solved by the receiving group.

For the learner-designed DebugIts, students again reported having the most difficulty interpreting the connection between the given code and circuit. One group wrote about challenges related to "trying to understand the code with the circuit," while another group was more specific, writing that the difficulties lay in "figuring out how the code effected [sic] the circuit." Thus, as with the instructor-designed DebugIts, students were forced to engage at the interface between the physical and digital elements of e-textiles through the design of their debugging challenges.

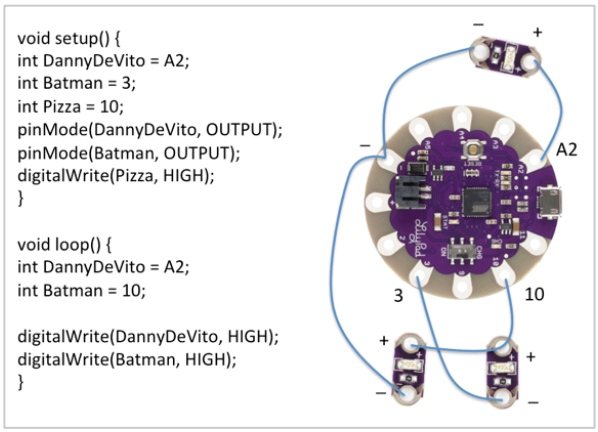


**Figure 7: Group 3's DebugIt with unconventional code and circuitry. The code (left) features redefined variables within the loop after already being defined earlier, while the circuit has a redundant ground pin and crossed (yet working) wires.**

## 5.3 Students Perceptions of DebugIt Activities

As a result of participating in the workshop, five of eight students felt that they learned about circuitry. Kevin[3] stated: "I can relate it to engineering," while Maria wrote about polarity: "I learned how to make sure (-)'s and (+)'s matter." Seven of eight students believed that the DebugIt activities and kit helped them learn coding. Anh stated that she learned "how it works and how to create it," while Maria pointed out how she gained understanding in "how little words matter a lot" when coding.

When asked about their experience with the complex reconstruction mats, most students (six of 10) chose fixing buggy code as the most helpful to their learning; Noah stated: "It was the hardest thing I did." In terms of creating DebugIts, student opinions were divided. Four of 10 students choose it as the least helpful to their learning, explaining: "I learned a little, but not the most out of this," "too easy," and "I really didn't know how to do

[3] This name and all following are pseudonyms.

it." On the other hand, four choose creating DebugIts as the most helpful to their learning. About solving and creating challenges, Servino stated: "Both of these were very interactive and helped us develop new techniques." Maria added that "It was cool being able to create my own challenge." Thus, even though students learned the most from the practice of debugging itself (i.e., fixing code), the creation of DebugIts potentially highlights an area where students felt the greatest opportunity to express themselves creatively within the context of debugging.

## 6 DISCUSSION

Our results illustrate the feasibility of our reconstruction kit in creating opportunities for rapid construction, deconstruction, and reconstruction with e-textiles. In designing activities for fixing and designing DebugIts, we allowed students to engage with troubleshooting both on and off-screen. In the discussion, we highlight both the affordances and limitations of our design, as well as suggestions gleaned from our prototype and testing for creating reconstruction kits for making at large.

### 6.1 Personalized Pathways of Productive Failure

One major affordance of the reconstruction kit is that its modularity creates opportunities for students to enter into personalized pathways of productive failure [13]. In other words, each reconstruction kit can yield multiple debugging challenges, all which can be customized to students' own abilities and desires. This differs from our earlier effort where we created the pre-sewn debugging challenge that had a set level of difficulty, and which could only be fixed once [8].

This customization afforded by the kit was evidenced by the different pathways that groups took through the activities. For instance, some groups took longer to grapple with basic concepts of circuitry and code and needed more time with the instructor DebugIts. When it became clear that groups were unable to finish on their own, we sometimes asked them to work on an even simpler task (e.g., get one LED to turn on) before moving back to their original challenge. Other groups, however, were able to finish challenges relatively independently. They tinkered and experimented with multiple solutions along the way, something evidenced by the high number of moves and alternate solutions they reported. In this way, the DebugIts provided a continuously personalizable ill-structured problem space for students to engage; that is, based on their own knowledge and assumptions, they could keep on 'failing' at debugging—tinkering and testing out their ideas—until they arrived at the desired solution.

### 6.2 Engaging Interfaces between On and Off Screen

Another main affordance of the reconstruction kit is how it enables students to work at the interface between the on and off-screen elements of e-textiles: the *digital* world of code and the *physical* world of the circuits. Debugging within hybrid computational spaces such as e-textiles and robotics is often difficult and complex since it involves not only knowing different domains (hardware design, computer programming) but also understanding the interrelationship or interface between these areas [12, 23]. This activity can become particularly overwhelming in practice; for instance, a non-functioning e-textiles or robotics project is more likely due to overlapping, multi-domain issues rather than one isolated problem or topic.

Our kit addresses this issue by creating opportunities to explicitly engage with this interface between digital and physical elements. Specifically, this was accomplished through the careful design of the DebugIts, which all purposefully addressed the interrelationship between code and circuitry in e-textiles. This was accomplished by setting up a system of only providing a buggy circuit or buggy code, along with a functioning complement (working code or working circuit) as a jumping off point for students. Thus, rather than encountering a complex, multi-domain problem all at once, groups were scaffolded into the practice, knowing that they had to refer to either the working circuit or code in order to understand what was broken in the other part. This structure therefore reinforced their understanding of the interrelationship between these elements, something that was highlighted by the numerous comments that students provided about this connection. In creating these pre-compartmentalized issues then, we scaffolded students into the practice of isolating parts of the system in order to understand the whole, something that we earlier highlighted as key component of computational thinking [24].

### 6.3 Designing Reconstruction Kits for Making

Many lessons about the design of reconstruction kits in general can be gleaned by looking at experiences of students who engaged with our prototype. One prominent issue that came up was the finicky nature of the pieces themselves. While we created connections that were easy to take apart (hooks and strips), this sometimes thwarted students in their efforts to creating working circuits; because they were so loose, they would accidentally just fall apart. This points to the need to strike a balance between easy deconstruction and secure connections in the design of these kits. Another issue we encountered was how the physical configuration of the pieces themselves sometimes constrained students' designs and understanding. As described earlier, the felt strips we used actually insulated the metallic thread such that crossed wires were not an issue. As a result, students did not come to understand the importance of how this can lead to short circuits—a common problem within regular e-textiles construction. Furthermore, the strips themselves tended to lead more easily to the creation of parallel circuits than independent circuits, something that might have caused students to understand the former better the latter. From this perspective, designers of any reconstruction kit should consider how the particular configuration of pieces might shape student understanding in unexpected ways.

Beyond these material concerns however, one of the most prominent lessons that can be learned from the development and the implementation of our prototype is how the kit can create opportunities for creative expression within the context of

debugging. Research has already pointed to the affordances of e-textiles production in creating spaces for aesthetic and narrative expression, often supporting personal identity, social connection and cultural relevance [3, 21, 22]. In this study, students lacked this opportunity since they were not asked to produce artifacts of their own. This is certainly a limitation of this study, and something we aim to address in future research. However, we argue that there was still room for creativity within context of the activities that we provided, namely within the process of problem solving. This can be seen within the alternate solutions students provided for the instructor DebugIts, as well as their own DebugIt designs, where they often presented unique and unusual solutions.

This is not to say that our workshop allowed for complete creative freedom though. For the purposes of making things accessible to students, we carefully outlined the steps that students needed to take in designing DebugIt challenges. In some respects, these constraints seemingly limited their creativity: one student, for instance, specifically asked whether he could add bugs to both the circuit and the code, something we did not allow since it did not conform to our structure (we only asked students to add bugs to either the circuit or the code). Future work looking at students' design of DebugIts might consider expanding upon this aspect of creative expression, perhaps drawing from research on youth designing games. Just as games ask others to solve a puzzle, DebugIts also have an audience that is supposed to provide a solution. The early designs of the Scratch DebugIt studio have already illustrated the potential of engaging learners not only in making projects but also in fixing them. Our findings further add to this area by considering students' *design* of debugging challenges within making as something that can become part of their learning, something that the development of reconstruction kits may afford.

## ACKNOWLEDGEMENTS

This work was supported by a grant (#1742140) from the National Science Foundation to Yasmin Kafai and Mike Eisenberg. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, University of Pennsylvania, or the University of Colorado, Boulder.

## REFERENCES

[1] Paulo Blikstein and Dennis Krannich. 2013. The Makers' Movement and FabLabs in Education: Experiences, Technologies, and Research. In *Proceedings of the 12th International Conference on Interaction Design and* Children (IDC '13). ACM, New York, NY, 613-616.

[2] Peter Brusilovsky. 1993. Program visualization as a debugging tool for novices. In *Proceedings of INTERACT'93 and CHI'93 conference.* ACM, New York, NY, 29-30.

[3] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, 423-432.

[4] Sharon McCoy Carver and Sally Clarke Risinger. 1987. Improving children's debugging skills. In *Empirical studies of programmers: Second Workshop*, Gary Olson and Sylvia Sheppard, and Elliot Soloway (Eds.). Ablex Publishing Corp., Norwood, NJ, 147-171

[5] Ryan Chmiel and Michael C. Loui. 2004. Debugging: from novice to expert. *ACM SIGCSE Bulletin* 36, 1 (March 2004), 17-21.

[6] Debug It!: 2010. http://scratched.gse.harvard.edu/resources/debug-itAccessed: 2017- 07- 27.

[7] Deborah A. Fields, Debora Lui, and Yasmin B. Kafai. 2017. Teaching Computational Thinking with Electronic Textiles: High school Teachers' Contextualizing Strategies in *Exploring Computer Science*. In *Conference Proceedings of 2017 International Conference on Computational Thinking Education* (CTE '17). The Education University of Hong Kong, Hong Kong, China, 67-72.

[8] Deborah Fields, Kristin 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* (FabLearn '16). ACM, New York, NY, 83-85.

[9] 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* (SIGITE '16). ACM, New York, NY, 148-154.

[10] Erica R. Halverson and Kimberly Sheridan. 2014. The maker movement in education. *Harvard Educational Review* 84, 4 (2014), 495-504.

[11] Yasmin B. Kafai, and Quinn Burke. 2014. *Connected code: Why children need to learn programming*. MIT Press, Cambride, MA.

[12] Yasmin B. Kafai, Deborah A. Fields, and Kristin Searle. 2012. Making the connections visible: Crafting, circuitry, and coding in high school e-textile. In *Textile Messages: Dispatches from the World of E-Textiles and Education*, Leah Buechley, Kylie Peppler, Michael Eisenberg, and Yasmin Kafai (Eds.). New Literacies and Digital Epistemologies. Volume 62. Peter Lang Publishing Group, New York, 85-94.

[13] Manu Kapur. 2008. Productive Failure. *Cognition and Instruction* 26, 3 (2008), 379-424.

[14] Breanne K. Litts, Yasmin B. Kafai, Kristin A. Searle, and Emily Dieckmeyer. 2016. Perceptions of Productive Failure in Design Projects: High School Students' Challenges in Making Electronic Textiles. In *Proceedings of International Conference of the Learning Sciences, Volume 2* (ICLS '16). International Society of the Learning Sciences, Singapore, 1041-1047.

[15] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. Debugging: a review of the literature from an educational perspective. *Computer Science Education* 18, 2 (2008), 67-92.

[16] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., New York.

[17] Roy D. Pea. 1986. Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research* 2, 1 (1986), 25-36.

[18] Kylie Peppler and Diane Glosson. 2013. Stitching circuits: Learning about circuitry through e-textile materials. *Journal of Science Education and Technology* 22, 5 (2013), 751-763.

[19] Mike Petrich, Karen Wilkinson, and Bronwyn Bevan. 2013. It looks like fun, but are they learning. In *Design, make, play: Growing the next generation of STEM innovators*, Margaret Honey and David E. Kanter (Eds.). Routledge, New York, 50-70.

[20] Mitch Resnick, and Brian Silverman. 2005. Some Reflections on Designing Construction Kits for Kids. In *Proceedings of the 4th International Conference on Interaction Design and Children* (IDC '05). ACM, New York, NY, 117-122.

[21] Kristin A. Searle and Yasmin B. Kafai. 2015. Culturally responsive making with American Indian girls: Bridging the identity gap in crafting and computing with electronic textiles. In *Proceedings of the Third Conference on GenderIT*. ACM, New York, NY, 9-16.

[22] Kristin A. Searle, Deborah A. Fields, Debora A. Lui, and Yasmin B. Kafai. Diversifying high school students' views about computing with electronic textiles. In *Proceedings of the tenth annual conference on International computing education research* (ICER '14). ACM, New York, NY, 75-82. ACM.

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

[24] Jeannette M. Wing. 2006. Computational thinking. *Communications of the ACM* 49, 3 (March 2006), 33-35.