Tensions in the Productivity in Design Task Tinkering - Fundamental

Ms. Gina M Quan, University of Maryland, College Park

Gina Quan is a doctoral candidate in Physics Education Research at the University of Maryland, College Park. She graduated in 2012 with a B.A. in Physics from the University of California, Berkeley. Her research interests include understanding community and identity formation, unpacking students' relationships to design, and cultivating institutional change. Ms. Quan is also a founding member of the Access Network, a research-practice community dedicated to fostering supportive communities in undergraduate physics departments, and an elected member of the Physics Education Research Leadership and Organizing Council (PERLOC).

Dr. Ayush Gupta, University of Maryland, College Park

Ayush Gupta is Research Assistant Professor in Physics and Keystone Instructor in the A. J. Clark School of Engineering at the University of Maryland. Broadly speaking he is interested in modeling learning and reasoning processes. In particular, he is attracted to fine-grained analysis of video data both from a microgenetic learning analysis methodology (drawing on knowledge in pieces) as well as interaction analysis methodology. He has been working on how learners' emotions are coupled with their conceptual and epistemological reasoning. He is also interested in developing models of the dynamics of categorizations (ontological) underlying students' reasoning in physics. Lately, he has been interested in engineering design thinking, how engineering students come to understand and practice design.

Tensions in the Productivity in Design Task
Tinkering - Fundamental

Introduction

Tinkering is an ad-hoc approach to a problem and involves the practice of manipulating objects to characterize and build knowledge about a particular system in an exploratory way, often with the goal of getting some product/idea to produce desired behavior.[1-5] Tinkering thus contrasts with more deliberate activity towards conceptual understanding of *how* some phenomenon works or more pre-planned approaches to design. Some researchers have argued that tinkering is an unproductive process because it does not always lead to progress and/or conceptual learning.[4,5] Others view it as productive for students' learning and for generation of novel solutions.[1-3] In this paper, we do a fine-timescale analysis of the process of tinkering to speak to this tension about the productivity (or unproductivity) of tinkering for novice designers and programmers. We claim that tinkering, or ad-hoc sense-making, *can* play a productive role in making progress towards design-activity goals. We argue for a more nuanced understanding of the benefits and drawbacks of tinkering.

Tinkering versus Deliberate Sensemaking

We define tinkering as both a process and an orientation. Processes of tinkering are often messy to identify, but we operationalize tinkering to be an ad-hoc trial and error.[1,5] Tinkering involves the rapid prototyping of ideas, and information gathered during each prototype drives subsequent trials. This is what Turkle and Papert describe as bricolage — a plan-as-you-go approach, which they contrast with a more formal, planned approach.[2] We contrast tinkering with deliberate sensemaking, which is a more systematic and planned activity with the goal of making sense of a phenomenon or system. The goal of tinkering, on the other hand, is to produce a product or outcome. This is not to say that conceptual sense-making and tinkering are mutually exclusive processes. However, in tinkering the goal of producing an outcome drives conceptual sense-making — that is, conceptual understanding happens only in the service of the outcome, rather than as an end in itself. This goal orientation in tinkering (or, at least the kinds of activities we are calling tinkering in this paper) contrasts other kinds of unstructured knowledge generation, such as play, or messing about,[6] where there is no clear goal.

Some researchers describe tinkering as a productive activity. In their study of students learning to program, Berland et. al. situate tinkering as an essential step between the initial exploration phase and refinement of ideas.[1] In a different vein, Turkle and Papert consider tinkering as a valid end-goal. They argue that some ways of knowing, such as bricolage, are more authentic in some situations for some people, and stress the value of multiple ways of knowing and learning ("epistemological pluralism" p. 161).[2] They also highlight the unique affordances of tinkered approaches, such as helping students move past roadblocks easier. Tinkering is also consistent with Roth's description of the design process.[3] He illustrates how a given design is often emergent through the process and context. In this framework, the artifact at a given moment is reflective of its' prior states and will influence future states.

Others describe tinkering as unproductive for the lack of clear aims and progress. In a study of novices and experts engaged in debugging, Law argues that tinkering did more harm than good, introducing additional bugs, and leading to more overcorrections than in planned approaches.[5]

Yeshno and Ben-Ari also suggested that trial and error is only useful if it leads to conceptual learning.[4]

In this paper, we add to this debate by providing empirical evidence that in some situations tinkering can play a productive role in problem solving. Through fine timescale analysis of unfolding events during tinkering activities in an engineering design learning environment, we show how tinkering can help students make progress towards their design goals and in some instances, even motivate their engagement in more systematic sense-making. We also suggest that there is more to be gained from tinkering than possibly conceptual skills. We do not mean to suggest that tinkering is universally productive or universally unproductive: rather, we think that through fine-grained analysis of episodes of tinkering activities embedded within a broader task, researchers can get a better handle at how to characterize productive or unproductive tinkering behavior. This, in turn, can help us generate instructional strategies towards scaffolding tinkering behaviors in the classroom.

In the following sections we describe the context of the learning environment, data collection, and analysis methodology. Then we present our analysis of two cases; in each case, a student design pair engages in tinkering behavior. In the first case, tinkering happens to set the stage for more conceptual sense-making. We highlight how tinkering both encouraged productive engineering practices and helped students engage in sense-making. In the second case, we describe an instance of less productive tinkering. Students self-generated and got stuck in an unachievable task, yet were still able to engage with the content in productive ways. We then present interview data from one student in the second case, to illustrate how tinkering may interact with students' emotional experiences. We suggest that regardless of whether or not students can complete a design goal, tinkering can help students engage in productive disciplinary practices.

Classroom Background

We designed and ran a project-based instructional module within Summer Girls, a day-camp for high school students hosted by the University of Maryland. The module was piloted in Summer 2013, and small modifications were made and implemented in Summer 2014. As part of the program, students learned to program Arduino (microcontroller) controlled robot-tanks (henceforth, Arduino-bot). Roughly 1-2 hours per day were dedicated to Arduino activities, while the rest of the time was spent on modern physics lectures, lab tours, and demonstrations. The Arduino classroom was structured to have a high level of student agency. Throughout the program, participants worked in groups of twos or threes through several open-ended Arduino design tasks before designing and completing a final project using Arduinos. Design tasks required students to program the Arduino-bot to perform some task such as detecting an obstacle, visually depicting distance from a wall, etc. The camp was co-taught by two instructors. Each day, there were 2-3 graduate student and undergraduate volunteers to help students with their projects. Students were also given a reference library of sample code, and were strongly encouraged to reference the internet.

Methods

Over two iterations of the summer camp, we collected interviews, coursework, and classroom videotapes of focal groups. Due to limited resources, we filmed one pair in the pilot year. In the second iteration, we filmed two pairs and two trios. Interviews occurred at the beginning of the

camp, beginning of the second week of camp, and after camp ended. Because interviews were voluntary, only two students completed all three interviews. We collected classroom videotapes of all Arduino activities, and some additional in-class activities. We also collected some written work, including daily written feedback, of all consenting participants.

The classroom data corpus was roughly chunked, and marked for focal episodes where students employed multiple strategies to solve a complex problem. For each focal episode, we categorized the students' epistemic goals (e.g. completing a task, understanding a concept) and created more detailed content logs in which we described the kinds of approaches employed or proposed. Our analysis attends to how tinkering played a role in students' design process, in attempt to understand how tinkering may or may not be a productive process for students. We look for evidence of particular student goals through speech, gestures and actions using interaction analysis.[7] We selected two focal episodes for this paper, to describe how the process of tinkering can support students in engaging in more in-depth sense-making. The interview data corpus was similarly chunked and marked for instances where students described tinkering-like processes. Those episodes were analyzed to characterize students' views of what they thought they were doing during design, and the role of tinkering-like processes. We pair one interview with the second classroom focal episode, to shed light on the student's perspective on the tinkering process. Data and preliminary analyses were presented at multiple University of Maryland Engineering Education research group meetings, to ensure consideration of multiple interpretations of the data and to sort out the interpretation supported by the largest fraction of data.

Hazel and Silver: Tinkering Leading to Conceptual Sensemaking

Our first episode begins when Hazel and Silver (pseudonyms) had just completed a short task in which they were asked to write a program to make the Arduino-bot move forward until it detected an obstacle/wall, and then make a right turn. Hazel and Silver initially did not program the Arduino-bot to stop after turning right, so after the Arduino-bot turns right it starts running over the keyboard until Silver grabs it. In response, they decide that they should make the robot stop after turning right. The goal of making the robot stop was not assigned in class, but many elements contributed to the reinforcement of the task as a goal.

We see the goal of stopping the Arduino-bot emerge through the constraints of the physical space: had Silver not grabbed the bot, it would have run off the table. When Silver suggested modifying the task to stop the bot, Hazel immediately took up the task and offered suggestions for how to make it stop. Later, other groups ask them if they had been able to make it stop, suggesting that other groups were also adding "stopping" to their task and thus reinforcing this goal for Silver and Hazel. The openness of the classroom culture in which students felt ownership over the project task also played a role in students feeling comfortable in modifying the task statement to include stopping — no group asked the teacher for permission to do this. In that sense, that the problem statement is emergent is some indication that students are in an exploratory, tinkering mode in working with the robots.

What follows in the next several minutes is Hazel and Silver systematically tinkering through a variety of strategies to make the robot stop. They first begin by adjusting the digital outputs,

which are set to HIGH.[1]

Hazel: We could just, we could also make it stop.
Silver: Oh, like after it gets to this distance.
[Both girls lean into the computer monitor][2]
Hazel: We could turn these from, like, LOW stop.. I just changed this one to LOW.
Silver: What was it before?
Hazel: It was HIGH. Cause if one (inaudible)[pointing to left tread] it makes it turn right.
Silver: Oh. So LOW means off.
Hazel: Yeah, same with the light, and brightness.
[Girls upload the program, and the bot moves forward, slows down, and runs into the box]
Hazel: It's now trying to stop. Oh, no, no
Silver: Oh, it's now turning LOW power to HIGH to just like- stop. Do you think that's in the PDF?
Hazel: Yeah
Silver: Yeah
[Girls open an Ardunio bot reference sheet]
Silver: It doesn't say how to stop.
Hazel: No, but that's a thing we can Google.

Here, tinkering supported extended engagement in the activity. They first try something that had worked in a prior LED task, in which rewriting a digital output from HIGH to LOW turned off an LED. They similarly change the digital settings of the motors, not knowing that LOW and HIGH correspond to the motors' directions, rather than speed. This strategy leverages knowledge from prior tasks to predict a solution, a strategy that we see as productive, even though in this particular situation it does not yield the desired result for Hazel and Silver. While drawing out that connection, they try to make sense of how motor treads work (getting information about the new system rather than simply applying knowledge from previous experience). Their dialogue and gestures give evidence that they see that turning a motor off will cause a corresponding tread to stop moving. After their first modification does not work, they spend a couple of minutes checking a reference guide and then search on Google for the solution, practices which were encouraged in class. We see this utilization of available resources, without getting too bogged down in one resource, as productive trial-and-error.

Three minutes later they isolate and execute individual lines of code (commenting out the other lines), to generate some knowledge about what individual commands do. We see the whole segment as tinkering because they do not attempt to draw out underlying principles of how the system works; at a later point in the episode, they even verbally acknowledge that they do not understand what individual functions do. The rapid testing of activities occur in the span of about seven minutes.

[1] HIGH & LOW: refer to the state of digital pin in the output mode. A command such as **digitalWrite(9,HIGH)** would set an output voltage of 5V at the digital pin 9, while the same command with LOW instead of HIGH Will set that digital pin to 0V.
[2] [] Indicate actions or gestures

One could argue that instead of tinkering, Hazel and Silver should have systematically parsed the code to make sense of it right from the start; they would have had better task success and better learned Arduino programming through that process. We contend this notion. Hazel's and Silver's activities reflect a recognition of the variety of resources at their disposal and a systematic walk through the resources to try and achieve their goal. At each stage, they expanded the scope of their investigation: first, getting feedback from manipulating the specific system (Arduino-bot), then searching resources provided by the instructors, and finally searching on the internet. Furthermore, their activities reflect a certain level of judgment and resource management: they did not get fixated on any path, but quickly judged if the path would be productive, and if not, they switched tactic. These are all important skills in the design process, and within authentic engineering practice are not trumped by the value of systematic processing of the code. The rapid movement from one idea to the next also reflects metacognitive time management; they do not get bogged down in any one strategy testing of multiple paths. They also do some broader exploration of the system, doing work to understand the relevant components of the robot and code.

We do not, however, deny that line-by-line processing of code is also a valuable epistemic practice. But it is also time-consuming and requires deferring the goal while one makes sense of the code before they are ready to manipulate it toward the desired goal. On the other hand, tinkering or looking up resources, if successful, can be a faster route to the solution. In that sense, if achieving a particular function is the primary objective (as it was in the case of Hazel and Silver), then starting as they did can be a productive strategy. What could make tinkering processes unproductive is if a student gets fixated for a long time on these earlier methods even when they aren't yielding desired outcome. Hazel and Silver, however, do not get fixated. Indeed, when the other strategies do not meet the desired solution (how to make the bot stop) in about seven minutes, we see Hazel and Silver switch to engaging in a more systematic analysis of the code.

> Hazel: Okay. So we know that, that HIGH was backwards. It is the reverse of what it's supposed to be.
> Silver: I think the distance part is messing it up now.
> Hazel: Okay, yeah.
> Silver: Maybe take out the distance part and see if it will go for, like a, certain amount of time, or a certain-
> Hazel: We have a delay here, I think that's what's causing things to be weird cause- can we have more than one (inaudible)? No.
> Hazel: So it takes a reading. It calls it for one second. It goes forward. It…

Here we see them begin synthesizing their observations into an understanding of what lines of code do. Hazel remarks, "So we know that, that HIGH was backwards." We then see Hazel parsing each line of the code in terms of its functionality, "So it takes a reading. It calls it for one second. It goes forward." However, it was while tinkering with the variables in the code that they explicitly problematized their lack of understand of certain functions. This awareness could have contributed to their getting to parsing the code to make sense of the functions. Thus, in some situations, tinkering can give students a sense for where they might focus a more deliberate analysis.

Bianca and Coral: Disciplinary Practices in Tinkering

The next clip highlights how tinkering can still engage students in productive disciplinary practices, even when the end-goal is not the most productive strategy. In the first week of Summer Girls, Coral and Bianca generate a goal of completing a 90 degree turn with the Arduino-bot. This kind of precise task is often difficult to do reliably with the Arduino-bot due to inconsistencies in Arduino-bot's motion over short timescales.

Coral and Bianca had just completed a task of detecting and avoiding an obstacle and had moved onto the next task of completing a "maze." The "maze" was a pathway of left and right turns, with "walls" made of wooden blocks, and was set up in the back of the classroom, and students were allowed to conduct multiple trials. Instructors had intended for students to adapt the obstacle avoidance code, using a closed-loop control strategy to identify walls with the distance sensor, and use logic control structures to determine steps. At the start of the task, Coral says that the maze likely has 90 degree turns, and suggests that they first find the delay to complete a 90 degree turn. This practice of breaking a larger task into small testable pieces is often a good design practice, but in this case, it ended up being part of an open-loop control strategy, where they pre-programmed each piece of the Arduino-bot's motion through the maze. In general, open-loop strategies can be more susceptible to slight variations in the physical setting as compared to sensor-based closed loop strategies that can adapt to the variations. We should note, however, that though the participants in the camp were introduced to sensors, they hadn't been explicitly instructed on the merits and demerits of open- versus closed-loop control strategies.

The clip begins with Coral and Bianca trying to get the bot to make discreet turns. Their code writes the Arduino-bot's outputs to the turn settings. In their first trial, the bot turns without stopping. They are surprised, and Coral immediately picks up the (attached) USB cable and holds it over the bot, so it does not run over the cable. After the first test, they realize they forgot a delay after the code that sets the bot to turn, so the code is continuously looping through the turn.

> Bianca: Woo! Whoa, Okay!
> Coral: Oh it's just turning around
> Coral: We forgot- Oh, we have to do a delay on it. I forgot. Let's try a delay of like one second just to see.

Coral suggests a delay of one second "just to see." Here, she seems to be inserting a placeholder value based on what she thinks is a reasonable delay. The hedge in her utterance "just to see" indicates that she is probably expecting to use this as a trial run to see how the system behaves (getting system-specific information) when a delay is added and that she will be refining that value later, either systematically or through trial and error. Because the next changes to the time delay become smaller and smaller, we argue that she was using a placeholder value, with the intention of narrowing in on the right amount. The statement also signals that she intends to use the information from the next trial to inform the next modification, demonstrating the tinkering nature of the activity. After uploading the new code, the Arduino-bot again spins around in circles, as there is no other code after each spin in the loop.

> Coral: It's a loop so it just keeps spinning.
> Bianca: Alright well.

Coral: So it spins for a second, and it reads it again and spins for another second. Infinitely. How do you make it stop? Oh you have to. Uhhh?
Bianca: Then we should probably do like, 'And then go forward.' Instead of turn.
Coral: Ah yeah I guess so.

They use the test as a way to generate knowledge about the system - Coral realizes that the spinning happens because the code runs through the turn and loops again. Bianca then modifies their goal to now include a right spin and forward motion, which is necessary for them to be able to see if they have a satisfactory spin. This instance is similar to what Turkle and Papert describe "a collaborative venture with the machine (p. 136)," treating the mistake as a part of one's navigation, rather than a bug. Bianca adds several more lines of code to have the bot move forward after it turns. The bot successfully loops through the turn and forward motion, even though the turn is not 90 degrees.

Coral: We have to try and figure out what the angle is so-
Bianca: It was a little bit more than 90 so
Coral: So let's say it's probably about like
Bianca: It's like 135, so we need it to be like 45 degrees less
Coral: Right. So change it to like-
Bianca: Should we change the delay to -
Coral: Oh yeah you change the delay to like, 750. Wanna try that?

In this segment, they move from one aspect of their goal, getting the bot to spin and move forward, to getting the timing of the spin. The pair then spends almost four minutes adjusting and re-adjusing the delay over several trials, making smaller and smaller changes. In one run, the bot turns 90 degrees on the first turn, but in the second turn, it turns more than 90 degrees. Bianca starts to notice an inconsistency in how the bot runs.

Bianca: Yeahh! That's like perfect. But how come when it does it the second time it doesn't?
Coral: I think it was the cord. I think the cord was pulling on it.
[Coral removes cord and runs again. The bot keeps looping through the turn and forward motion]
Coral: Oh that's a tiny bit more than 90 I think.
Bianca: Why?
Coral: See? Now that was 90! And that was a tiny bit more than 90?
Bianca: Why is it so inconsistent?
Coral: So it went from like here to like
Bianca: Yeah.
Coral: It probably like, it's probably 775.
Bianca: Or maybe we should make it like 780.

In this clip, Bianca notices that the bot doesn't turn for the same amount in each loop. Coral at first suggests taking out the USB cord, attending to how the physical setup could be causing variability in the motion. Bianca makes a bid to understand why it's so inconsistent, but they revert back to fine-tuning the delay. The period of rapid testing ends when Coral decides to look at the maze. She briefly consults with a classroom helper.

Coral: We're figuring out 90 degree turns
Helper: Awesome.
Coral: I hope the maze has 90 degree turns.
Bianca: Yeah that would be, it should be.
Coral: Is the maze up? Do you know?
Helper: I think so.
[Bianca uploads and runs another iteration]
Coral: It's a little bit-
Bianca: Why?
Coral: Cause like if starts like straight, it's pretty close.
Bianca: It's still gonna run into the wall though.
Coral: Well for now, we should probably actually like look at the maze. I'm gonna go-
[Coral gets up]
[Bianca types and tests new values on her own silently]

At the end of this clip, Coral goes to look at the maze while Bianca continues to try to fine-tune the delay. Revisiting and adjusting the problem while designing solutions is a productive practice.[8] Coral seems to be making a bid at the end for getting the turn "pretty close," because it might still go through the maze. Their approach to the maze ends up being an open-loop set of instructions to turn and go straight, rather than developing an algorithm using a distance sensor. Only toward the end of the task, they incorporate the distance sensor into the first turn and the end of the maze.

In some ways, their goal and approach to the task was unproductive. In it, Coral and Bianca are trying to fine-tune a time delay to get the Arduino-bot to make 90 degree turns. This goal is nearly impossible to accomplish based on the inconsistencies of the bot, though they do not know that when they start to work toward it. Despite Bianca noting the inconsistency of bot's turns, they still stay stuck on trying to accomplish this goal, rather than shift strategies. We suggest that the unproductive aspects of their activity lay in their goal choice; however, tinkering in itself is neither productive nor unproductive.

Still, this episode shows how engagement in tinkering enables them to engage in good design practices. They engage in multiple troubleshooting strategies, without getting too bogged down. For example, while working on the subgoal, they start their delay by using placeholder values, to see if their code generally does what they want it to. They also adapt their goal to have the Arduino-bot turn and run forward, based on information gathered from the system in their tinkering. This kind of goal adaptation frames their non-working code as a building block, rather than a mistake. They also try reducing error by taking into account physical features of the system (moving the USB cord) and modifying the code itself.

Valuing Tinkering for Affective Engagement

Now we turn to excerpts from our interview with Coral to illustrate how tinkering is also an emotional experience for some students; and their emotions can be coupled with their sense of whether the tinkering was productive or not. Coral describes tinkering-like practices as productive for design, and sees managing frustration as a part of tinkering. In an interview in the middle of camp, Coral discussed the nature of design, and how design requires frequent testing

of different solutions. At the time of the interview, Coral and Bianca had just begun their final project. Their project, a walking and dancing baby, required coordination of two separate Arduino-bots as the feet of the baby.

> Interviewer: How are you ensuring that the two robots will work together?
> Coral: So, we do have two programs for the two. But we know that from today we saw that one seemed to be moving a little bit faster than the other. So we were thinking of trying it just at a lower motor speed or possibly changing out the batteries of the one that was moving slower since we have been using those a lot and the other ones were like new batteries. So we thought about that and then also, just testing, a lot a lot of testing. and like slightly altering the program here but not too much where it'll make a drastic change and you have to alter the other one and yeah.
> Interviewer: So like, making little changes, seeing how it works, making a little more changes, seeing how it works.
> Coral: Yeah. And just like, not getting frustrated, being like, this is going to be difficult to move the two and we both understand that, so it's just fun and go from there.

Though Coral doesn't explicitly say "tinkering," her description of testing aligns with tinkering. Testing itself isn't necessarily tinkering; testing could also be part of more systematic data collection. We suggest that to her, testing is what we call tinkering, because testing occurs in the pursuit of accomplishing a goal, rather than drawing out underlying concepts. Her description of testing also explicitly includes more than one strategy, lowering speed and changing batteries, and using multiple strategies is also a feature of tinkering. Coral brings out managing frustration as an important component of doing design. She elaborates more on frustration.

> Interviewer: What happens when things aren't working? You're feeling frustrated?
> Coral: I think it's definitely like, you're going to get frustrated and I've done enough with robotics to know that it's a frustrating task sometimes. But you just have to kinda know that like, if I don't change it, it's not going to change, and if you want that end goal, or if you want it to accomplish what you want it to accomplish, it's just gonna take time. And testing different things, trying out different values, different codes maybe, different ideas, taking a second to like, just leave it and then just letting your mind play around with different ideas and just stepping away from the project for a second, and coming back to it. So, I mean, I think it's one of those things where, I do get frustrated but I don't think it's overwhelming where I ever really feel like 'Okay, that's it, I'm done.'

We also see a little more of what that testing process looks like for Coral. "Testing different things, trying out different values, different codes maybe, different ideas," reflects using multiple strategies to achieve a result. To Coral, this process is not only helpful, she sees it as an integral part of doing design. She accepts frustration as part of the design process and says that part of not letting the frustration turn into an obstacle to progress is to keep an eye on the goal you want to achieve and taking a break when needed. We can also see Coral as suggesting that the multiple quick moves that tinkering allows, including making quick changes and trying out new ideas could potentially alleviate frustration when something isn't working. The evidence to fully support this interpretation is thin, but points to the need for investigating how students' tinkering practices are coupled with their local emotions, which could impact their engagement with design.

Discussion

Though tinkering may not lead to generalizable content learning, we argue that it often has value as an engineering disciplinary practice. This work sheds light on how tinkering emerges in the design process, the ways in which tinkering might be productive for design, and how tinkering might impact student engagement.

In the case of Hazel and Silver, tinkering was a productive part of their process and helped them engage in more systematic unpacking of concepts. While tinkering, they utilized resources without getting too bogged down in one strategy, used multiple iterations to build specific knowledge about the system, and drew connections across tasks. They also identified aspects of code that they did not understand, and later did more systematic analysis of those areas.

Bianca and Coral tinkered through a self-generated goal that was part of a larger, unproductive open-loop control strategy. They still engaged in many good practices while tinkering. For example, they adjusted the goal based on the system, used placeholder values, and revisited the problem in the process of designing a solution. Interview data with Coral also pairs tinkering with positive affect related to lowered frustration, and some acceptance of frustration as being part of the tinkering process. The example of Bianca and Coral raises questions about what to do when students engage in productive practices yet are following an overall unproductive strategy. One could imagine steering students toward the more productive or more sophisticated strategy (such as a sensor-based closed-loop control strategy in the case of Coral and Bianca), and while we can envision that working out well, we can also envision situations in which such an intervention might lead students down the path of just doing what the instructor tells them to do. The specifics of enacting such an instructional intervention would matter greatly in how subsequent student action plays out. This example draws our attention to the different grain-sizes at which we might attend to when valuing or critiquing students' design thinking.

To summarize, we argue that students' tinkering behaviors can have a productive role in the engineering design classroom and more research is needed on understanding the various ways in which students take up tinkering during engineering design activities.

Acknowledgements

We are grateful to the participants in the Summer Girls program for allowing us to videotape their classroom participation and for volunteering for interviews. We thank Andrew Elby for his feedback on earlier drafts of this work. We thank members of the University of Maryland Engineering Education Group (blog.umd.edu/eerg) and Physics Education Group (ter.ps/perg) for valuable discussions around this data in group meetings.

This material is based upon work supported by the National Science Foundation under Grant Numbers DRL-0733613 and DUE-1245590.

References

1. Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, *22*(4), 564-599.
2. Turkle, S., & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs*, 128-157.
3. Roth, W. M. (1996). Art and artifact of children's designing: A situated cognition perspective. *Journal of the Learning Sciences*, 5(2), 129-166.
4. Yeshno, T., & Ben-Ari, M. (2001). Salvation for bricoleurs. In Proceedings of the Thirteenth Annual Workshop of the Psychology of Programming Interest Group, Bournemouth, UK (pp. 225-235).
5. Law, L. C. (1998). A situated cognition view about the effects of planning and authorship on computer program debugging. *Behaviour & Information Technology*, *17*(6), 325-337.
6. Hawkins, D. (1965). Messing about in science. InThe ESS Reader (pp. 37–44). Newton, MA: Elementary Science Study.
7. Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *Journal of the Learning Sciences*, 4(1), 39-103.
8. Dorst, K., & Cross, N. (2001). Creativity in the design process: co-evolution of problem–solution. Design studies, 22(5), 425-437.