Epistemological Pluralism for Diversifying Preservice Early Childhood Teachers' Programming Experience

ChanMin Kim, Ph.D. | cmk604@psu.edu | 814-865-9919
The Pennsylvania State University
University Park, PA 16802 USA

Brian R. Belland, Ph.D.
The Pennsylvania State University
University Park, PA, USA

Duygu Umutlu, Ph.D.
Bahcesehir University
Istanbul, Turkey

Abstract

Research indicates that computer programming in a bricolage manner is equally strong as structure programming. In this study, we investigated how and why 26 preservice, early childhood teachers learning to program employed diverse approaches to programming. Data included classroom videos, interviews, written reflections, submitted code, and questionnaires. Analysis involved open and axial coding. Findings included (a) all tinkered through trial and error but this does not mean that analytical means were never used, (b) divide-and-conquer was practiced, (c) analytical means were often used in locating the bug whereas tinkering was used mostly in fixing the bug, (d) unnoticing when/where to tinker compromised the programming goal, and (e) robot programming was perceived as creative, artistic, and playful.

Purpose

The goal of this study is to examine how participants engage in programming and debugging to figure out how the observed approaches could be leveraged to facilitate their learning (a) to program in bricolage and structure styles and also (b) of culturally responsive pedagogy in computing.

Perspectives

Epistemological pluralism refers to a doctrine "accepting the validity of multiple ways of knowing and thinking" (Turkle & Papert, 1990, p. 129). It is grounded in ontological pluralism accepting multiple kinds or modes of being (Turner, 2010). Epistemological pluralism has been revisited in recent STEM education research, especially when recognizing diverse ways of programming (Berland, Martin, Benton, Smith, & Davis, 2013; Fields, Kafai, Nakajima, & Goode, 2017; Searle, Fields, Lui, & Kafai, 2014). In Turkle and Papert (1990), students in a computer science course who used a bricoleur style in Logo programming completed the course as successfully as the ones who used a structured planner style. Bricoleur scientists do "science of the concrete" by (re)arranging and (re)negotiating with the materials at hand (Turkle & Papert,

1990, p. 7) and use events to create structures (Lévi-Strauss, 1966). In contrast, structured planner scientists use structures to create events, move hierarchically, and do science of the abstract (Lévi-Strauss, 1966). As such, structured planner programmers (structure programmers hereafter) value hierarchy and abstraction within computational objects, and use a top-down approach, dissecting the programming problems into subparts and addressing them systematically (Turkle & Papert, 1990). Bricoleur programmers value relationships with and concreteness of computational objects, and use a negotiational approach, tinkering through constant rearrangement (Turkle & Papert, 1990).

Despite the dominant culture in computer science privileging structure programming, the quality of product from bricoleur programming was as good as one from structure programming (Rose, 2016; Turkle & Papert, 1990). Turkle and Papert (1990) used an analogy of writing to programming: writing an article without an outline is as valued in the community of writers as writing with an outline; quality articles produced without an outline are not viewed as immature, inferior, or countercultural.

Preservice teachers who learned programming in a step-by-step systematic, analytic ways exhibited improved logical thinking as well as growth in computational thinking and interests in computer science (B. Kim, Kim, & Kim, 2013). Also considering the benefits of logical programming, especially as for debugging, argued in (C. Kim, Yuan, Vasconcelos, Shin, & Hill, 2018), pluralistic learning to program (through bricoleur *and* structure programming) could prepare future teachers to uncenter either one style of programming and actually use logic "on the tap not on top" (Turkle & Papert, 1990, p. 133).

The following research questions guided the study: How do preservice early childhood teachers program robots? In what ways do they debug?

Methods

Research Design

We used an ethnographic study (Goodman, 1988; Jurasaite-Harbison & Rex, 2010; Rozelle & Wilson, 2012; Zembylas, 2004) to examine the process of robot programming and debugging. Specifically, we used multiple data sources to investigate participants' conversations and actions and reasoning behind them.

Participants

Participants included 26 early childhood education preservice teachers enrolled in a course on integrating the arts in early childhood education in a large university in the United States. Twenty three indicated low or no knowledge of programming whereas three noted having intermediate programming knowledge. The average age was 21. Three of them were Hispanic or Latinx and the rest were Caucasians.

Robot Programming Unit

During Week 1 of the unit, participants were introduced to a sample lesson in which an Ozobot navigated to the vegetable section of a supermarket. They then studied the code that made it do so, and worked on optimizing the code. They then tried out the lesson in their preschool field experience placement. During Week 2, they reflected on teaching the lesson in field experience, completed coding challenges, designed a new lesson, and taught the new lesson

in their preschool field experience. During Week 3, they reflected on teaching and learned other robotics topics.

Data Collection and Analysis

Data sources were classroom recordings (using four video cameras), interviews, reflections, artifact collection, and questionnaires. In total, 1680 minutes of classroom videos, 120 minutes of interviews, 77 reflections, 13 team lesson designs, 13 team code submissions, and 2 questionnaires were collected.

Classroom videos and interviews were transcribed, handwritten reflections were digitized, and questionnaire responses were analyzed descriptively.

Qualitative analysis was performed using Nvivo 12. The first author read transcripts, performed open coding, and created an initial coding scheme based on open coding and relevant literature on programming and debugging, epistemological pluralism, bricolage programming, and structure programming. The third author used the initial coding scheme to code one participant's interview, reflections, and classroom video. The first and second authors reviewed and discussed the coding, and revised the coding scheme with the third author. The third author applied the revised coding scheme to another set of data, and discussed the process and revised the coding scheme with the first author. Then, the third author coded one more set of data, and the three researchers finalized the coding scheme based on the third coding.

Findings

The following five assertations were developed. Only four participants' data are used to illustrate due to space limitations.

All tinkered through trial and error but this does not mean that analytical means were never used

Programming in trial and error manners was observed and reported in video recordings, interviews, and reflections. For example, Alice said to her partner in the classroom video, "You do like five steps. Do you think it's enough? Maybe? Yeah. We could do a little guessing. Guess and check is my best friend." She reported that she was ultimately successful using guess-and-check during her interview: "It took a few tries to get it to go perfectly because you know, it's a sort of hit or miss sometimes when you're coding…but it ended up working out." Along with tinkering, analytical means were also used (see assertions below); that is, big BRICOLAGE programming and small structure programming coexisted.

Figure 1 show Alice's code in which three kinds of blocks (movement, light effects, and loop) were used in sequential and repetition control structures.

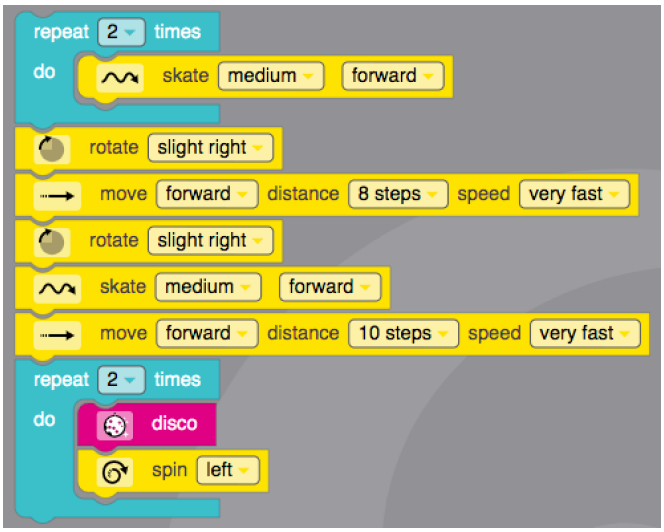

Figure 1. Alice's Ozoblockly Code

Divide-and-conquer was practiced

A structure programming strategy (Turkle & Papert, 1990), divide-and-conquer was used in conjunction with bricolage programming. For example, Nancy described her programming process in her interview: "Whenever I got home and I was able to just to really focus, I broke the code down into simpler steps and program that I was about to do each command before adding the next cycle of programming." Her debugging process also included a systematic approach:

> "I would go through and I would break the code down…until I got to the piece that was problem. I would fix that and then just reload the rest of the code and reprogram I was about to see if it fixed the problem that I thought I was having."

At the same time, Nancy performed trial-and-error in her classroom videos. Bricolage programming and structure programming were alternated within a task of programming. This suggests that structure programming is not an exclusive practice.

Figure 2 shows Nancy's code in which four kinds of blocks (movement, light effects, loop, and timing) were used in sequential and repetition control structures.

Figure 2. Nancy's Ozoblockly code

Analytical means were often used in *locating* the bug whereas tinkering was used mostly in *fixing* the bug

Instead of a quick guess immediately before a check, Rosie analyzed observed data with logical comparisons:

> "I looked first at the code to make sure that, you know, if he [Ozobot] is traveling up 10 steps turning right over eight steps, you turn. I was making sure that when he, after the u turn, he was still going forward eight steps instead of 10 making a left turn since he had just made a right turn to go back this way. And then 10 steps like making sure that I use the same numbers and then the opposite turns that way he would go back to where it came from and then I checked that it was all correct and then I looked at the OzoBot and had the instructor come help me calibrate it on the iPad cause I was thinking, okay maybe he's just not calibrated right…we tried to calibrate him like two or three times and run it again and it just still wasn't working."

Similarly, Alice reports an analytical process she used when trying to locate the bug:

> "We re-watched the Ozobot first. We definitely watched what the Ozobot was doing and made note of exactly where we thought it went wrong. And then we went back and looked at the code. We definitely looked at the code second after re-watching kind of what it was doing."

When attempting to fix the bug, interestingly, Alice tinkered: "we fixed a lot of the issues [by] going in and making little adjustments."

Failure to notice when/where to tinker compromised the programming goal

Rosie had a problem with positioning her Ozobot on a map; that is, the departure point that she placed the Ozobot had to be fixed to complete the code correctly. Instead of tinkering with re-positioning the Ozobot, she simplified her original code:

> "But the Ozobot wouldn't turn exactly ninety degrees even after I entered [Ozoblockly] Level 4 on the computer or whatever. And you can put the exact measurement you want them to turn. And he [Ozobot] still wasn't turning just ninety degrees and going over, he would, it would be a little different every time. So just the way he turned made it to where I could only do like an L and then he'd stop and then I've just replaced that object and he'd run the same path every time because it was more consistent than having him go to three different stops consecutively."

In Rosie's final code, only two kinds of blocks (movement and light effects) in Level 3 were used, which included only sequential control structure (see Figure 3).

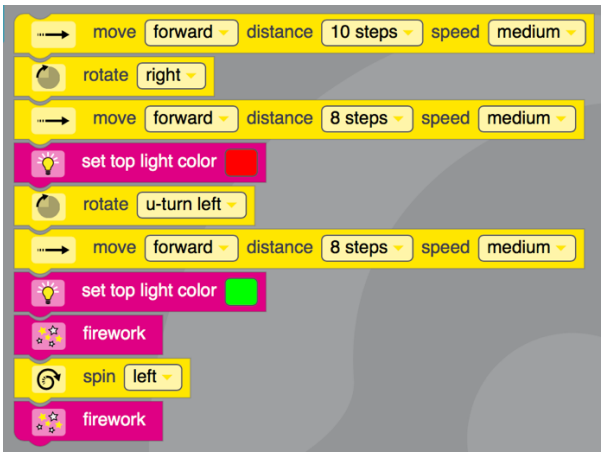

Figure 3. Rosie's Ozoblockly code

Robot programming was perceived as creative, artistic, and playful

> Alice wanted to engage in more playful experimentation with robot programming: "I just want to mess with this and see what I can make it do. And I guess I kind of wish that maybe we had a bit more time to just sort of mess with it on our own, um, in the class. Um, and just see, you know, program it to do sort of whatever we want just for like, just for a few, like a small portion of the lesson just so that we could sort of have our little like fun with it too I guess. Cause it was, they were just so fun to play with. When we had to make the lesson plan, we couldn't really just have fun. We had to really stick to the plan and, you know, be strategic about it."

Even though Alice changed her programming goal from the lesson plan, the whole process of programming was too structured to her. This may be because participants considered block-based coding more artistic than systematic as hinted in Nancy's comments in the interview: "… so that way she could give me the creative space to actually program the Ozobot and to do everything that was kind of artistic."

Conclusion and Scholarly Significance

The mixed use of bricolage programming and structure programming found in this study among early childhood preservice teachers is significant in that it contradicts the common conception that females most often use a bricolage approach (e.g., Searle et al., 2014). While bricolage was more used than structure programming in this study, the alternated use of both styles especially during debugging is an unprecedented finding, to our best knowledge

Another intriguing finding is that tinkering could help with keeping original programming goals. Tinkering by its definition is not a goal-oriented exploration (Berland et al., 2013). However, tinkering was used to achieve goals, and when unused, the goal was altered.

[Word Count: 2000]

Acknowledgements

This research is supported by grants 1927595 and 1906059 from the National Science Foundation (USA). Any opinions, findings, or conclusions are those of the authors and do not necessarily represent official positions of the National Science Foundation.

References

Berland, M., Martin, T., Benton, T., Smith, C. P., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, *22*(4), 564–599. https://doi.org/10.1080/10508406.2013.836655

Fields, D. A., Kafai, Y. B., Nakajima, T., & Goode, J. (2017). Teaching practices for making e-textiles in high school computing classrooms. *Proceedings of the 7th Annual Conference on Creativity and Fabrication in Education*, 5:1–5:8. https://doi.org/10.1145/3141798.3141804

Goodman, J. (1988). Constructing a practical philosophy of teaching: A study of preservice teachers' professional perspectives. *Teaching and Teacher Education*, *4*(2), 121–137. https://doi.org/10.1016/0742-051X(88)90013-3

Jurasaite-Harbison, E., & Rex, L. A. (2010). School cultures as contexts for informal teacher learning. *Teaching and Teacher Education*, *26*(2), 267–277. https://doi.org/10.1016/j.tate.2009.03.012

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, *49*(4), 437–459. https://doi.org/10.2190/EC.49.4.b

Kim, C., Yuan, J., Vasconcelos, L., Shin, M., & Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, *46*(5), 767–787. https://doi.org/10.1007/s11251-018-9453-5

Lévi-Strauss, C. (1966). *The savage mind*. Chicago: University of Chicago Press.

Rose, S. (2016). Bricolage programming and problem solving ability in young children: An exploratory study. *European Conference on Games Based Learning; Reading*, 914–921. Retrieved from http://search.proquest.com/docview/1859715060/abstract/F8840583576A4E48PQ/1

Rozelle, J. J., & Wilson, S. M. (2012). Opening the black box of field experiences: How cooperating teachers' beliefs and practices shape student teachers' beliefs and practices. *Teaching and Teacher Education*, *28*(8), 1196–1205. https://doi.org/10.1016/j.tate.2012.07.008

Searle, K. A., Fields, D. A., Lui, D. A., & Kafai, Y. B. (2014). Diversifying high school students' views about computing with electronic textiles. *Proceedings of the Tenth Annual Conference on International Computing Education Research*, 75–82. https://doi.org/10.1145/2632320.2632352

Turkle, S., & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs; Chicago*, *16*(1), 128.

Turner, J. (2010). Ontological pluralism. *The Journal of Philosophy*, *107*(1), 5–34.
Zembylas, M. (2004). The emotional characteristics of teaching: An ethnographic study of one teacher. *Teaching and Teacher Education*, *20*(2), 185–201. https://doi.org/10.1016/j.tate.2003.09.008