Programming Language Knowledge Transfer that Teachers Observe in their Classrooms

Jennifer Houchins
jhouchi@wested.org
WestEd
San Francisco, CA, USA

Rosalind Owen
rowen@wested.org
WestEd
San Francisco, CA, USA

Bryan Matlen
bmatlen@wested.org
WestEd
WestEd, CA, USA

Yvonne Kao
ykao@wested.org
WestEd
San Francisco, CA, USA

ABSTRACT

There has been significant progress in increasing the access to computing education for many K-12 students, including states adopting computer science (CS) standards and/or requiring CS courses. This includes the creation of block-based programming languages to make programming more accessible to younger students. Despite this progress, a new challenge has emerged: Students often struggle to transfer conceptual knowledge when transitioning to a new programming language (e.g., transitioning to a text-based programming after learning a block-based programming language). This poster presents the results of teacher interviews regarding the examples of knowledge transfer they observe in their classrooms. These interviews are part of an overarching project that aims to address the challenge of knowledge transfer between programming languages by developing a framework to support such transfer and deliver curricular supports that can be used to aid students' productive knowledge transfer between programming languages.

ACM Reference Format:
Jennifer Houchins, Rosalind Owen, Bryan Matlen, and Yvonne Kao. 2024. Programming Language Knowledge Transfer that Teachers Observe in their Classrooms. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635570

1 INTRODUCTION

Computing education literature contains studies examining how students transfer knowledge from block-based to text-based programming languages. One such study found little quantitative difference between the assessment scores between students who had prior Scratch block-based programming experience and those that did not [1]. However, the authors did note that students with prior Scratch experience employed Scratch-specific patterns in their text-based programs. This result suggests that some transfer did occur. Likewise, other studies have shown evidence of positive knowledge transfer from Scratch to text-based programming languages [3, 5]. The literature also contains studies which identify instances of positive transfer of knowledge between text-based languages [3, 5, 7] as well as documenting a lack of transfer [10].

Though these examples show considerable efforts, there has been little work regarding how teachers understand the problem of transfer and how they support students' transition from one programming language to another. Those studies that have examined ways of supporting students in transferring their programming knowledge have largely focused on the development of instructional materials [4] or programming environments [6, 9]. We believe a first step to understanding how and when students successfully transfer their programming knowledge is to understand how and when teachers observe transfer occurring in their classrooms, what explicit connections they make between programming languages, and what resources they are using to do so.

2 RESEARCH DESIGN

An overarching aim for this work is to help students from diverse backgrounds build on prior conceptual knowledge as they progress through a CS course sequence that involves multiple programming languages. The project is conducting a longitudinal study of students' progression through such a course sequence. To identify the ways in which students transfer knowledge from one programming language to another, we first need to understand how and when teachers observe transfer of programming knowledge, what explicit connections they make during their CS instruction, and what resources they are using to support such transfer. Therefore, we conducted interviews with middle and high school teachers (grades 6-12) who have experience teaching both block-based and text-based programming languages. The following research questions framed our study:

(1) What is the teacher's experience and coding background? [*Background*]
(2) What examples of transfer have CS teachers witnessed in their classrooms as their students learn a new programming language? [*Perceptions of Transfer during Learning*]
(3) How do CS teachers currently support students in transferring knowledge from one language to the next? [*Supporting Transfer*]

Eight teachers who self-identified as having experience teaching both Scratch and one other text-based language were recruited for

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635570

interviews. The interviews lasted for 1 hour and were conducted virtually via Zoom. The interview recordings were transcribed, and two members of the research team carried out a thematic analysis [2] of the transcripts. After the first pass of identifying themes, the two researchers reached agreement on refined themes that were grouped into subcategories such as *Spontaneous Transfer*, *Productive Transfer Resources*, or *Curricular Supports of Transfer*. This poster will include all subcategories, their descriptions, and examples from the transcribed interviews.

3 PRELIMINARY FINDINGS

Analysis of teacher interviews revealed that when teachers do make explicit connections between two programming languages, it is most often at the beginning of a new course. This depends on whether or not they know their students' prior programming knowledge. This result was only exhibited for three of the eight teachers who taught multiple CS courses and encountered the same students in more than one course along the curriculum pathway. These teachers indicated that they assumed students had no prior programming experience or that their students expressed having no prior programming experience when they were asked in class. However, the other five teachers noted that, most of the time, their instructional approach is to assume students have no prior programming knowledge. One teacher even suggested this was by design as an instructional technique learned from another content area, "*I assume kids usually don't have the prior knowledge because it was a technique I learned from math.*" Identifying students' prior programming exposure appears to be easier when teachers are at smaller schools and teaching multiple CS courses in the curriculum pathway.

Three of the teachers we interviewed also revealed that they primarily focus on differences in syntax when they are making explicit connections between multiple programming languages. They indicated that these connections were generally in the conceptual context of loops and other control structures, variables, and data structures. One teacher also suggested that they had to make their own classroom resources to show these connections, "*I literally had a PowerPoint and I broke it down to like, Hey, here's one major difference between the two.*"

Finally, four of the teachers interviewed noted that their students sometimes only received instruction in one programming language. These teachers explained that this is a result of students simply fulfilling a credit requirement for their particular curriculum pathway and/or to graduate. However, they perceived this as an issue that could impact knowledge transfer between programming languages because a student might take Scratch during their middle grade years (6-8) and not take another CS course until later in their high school course of study (grades 11-12).

4 CONCLUSION & FUTURE WORK

The study presented in this poster revealed important avenues for moving toward supporting students' programming knowledge transfer via teacher interviews. First, an overall lack of explicit curricular connections between multiple programming languages due to teachers' assumed or perceived potential of students having little or no prior programming experience appears to be at issue. This is an issue that most core academic disciplines do not experience as teachers have an understanding of students' prior knowledge from their course progressions. This strongly suggests a need for a standardized, pedagogically useful learning trajectory for every student on the CS curriculum progression. Further, development of such a learning trajectory could move the needle toward CS being a core discipline in every state with requirements at every grade level.

Additionally, when teachers do make explicit curricular connections between programming languages they tend to largely focus on syntactic differences rather than deeper conceptual knowledge. We believe that this is a missed opportunity for teachers to draw on and take advantage of the prior programming knowledge that their students bring to the classroom. For example, instructional guidance that employs tools like analogical theory [8] could prove to help conceptual knowledge transfer well. The next steps in this project will focus on avenues such as these to drive our work toward a CS-specific theory of knowledge transfer between programming languages. We will then use that theory to implement instructional scaffolds to aid teachers in making more explicit connections and garner the transfer of deeper conceptual knowledge.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through the EHR Core Research in STEM Learning and Learning Environments program (Award #2201209). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From scratch to "real" programming. *ACM Transactions on Computing Education (TOCE)* 14, 4 (2015), 1–15.

[2] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[3] Marcos J Gomez, Marco Moresi, and Luciana Benotti. 2019. Text-based programming in elementary school: a comparative study of programming abilities in children with and without block-based experience. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 402–408.

[4] Shuchi Grover. 2021. Teaching and assessing for transfer from block-to-text programming in middle school computer science. *Transfer of learning: Progressive perspectives for mathematics education and related fields* (2021), 251–276.

[5] Shuchi Grover, Roy Pea, and Stephen Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer science education* 25, 2 (2015), 199–237.

[6] Michael Kölling, Neil CC Brown, and Amjad Altadmri. 2015. Frame-based editing: Easing the transition from blocks to text-based programming. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. 29–38.

[7] Monika Mladenović, Žana Žanko, Andrina Granić, and Andrina Granić. [n. d.]. Mediated transfer: From text to blocks and back. 29 ([n. d.]), 100279. https://doi.org/10.1016/j.ijcci.2021.100279 MAG ID: 3134228633.

[8] Dario D Salvucci and John R Anderson. 2001. Integrating analogical mapping and general problem solving: the path-mapping theory. *Cognitive Science* 25, 1 (2001), 67–110.

[9] Nischal Shrestha, Titus Barik, and Chris Parnin. 2018. It's like python but: Towards supporting transfer of programming language knowledge. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 177–185.

[10] David Weintrop and Uri Wilensky. [n. d.]. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. 142 ([n. d.]), 103646. https://doi.org/10.1016/j.compedu.2019.103646 MAG ID: 2964965801.

Received 13 October 2023; Accepted 13 November 2023

Programming Language Knowledge Transfer that Teachers Observe in their Classrooms

Research Questions

We aim to identify the ways in which students transfer knowledge from one programming language to another. Our study is framed by the following questions:

- What is the teacher's experience and coding background? [*Teacher's Background*]
- What examples of transfer have CS teachers witnessed in their classrooms as their students learn a new programming language? [*Perceptions of Transfer during Learning*]
- How do CS teachers currently support students in transferring knowledge from one language to the next? [*Supporting Transfer*]

Methodology

- Purposeful recruitment of teachers who have taught both Scratch and at least one text-based programming language (***n = 8***)
- Conducted 1-hour semi-structured interviews virtually via Zoom
- Interviews were recorded to assist with note-taking
- Thematic analysis was carried out on the transcripts of the interviews.
- Refined themes emerging from the analysis were grouped into subcategories: ***Spontaneous Transfer, Productive Transfer Resources, or Curricular Supports of Transfer***

Findings

- When teachers do make explicit connections between two programming languages, it is most often at the beginning of a new course
  - This is dependent on knowing students' prior programming knowledge
  - Otherwise, teachers assume students' have no prior knowledge
- Explicit connections between languages tends to focus on syntax
- Students may only learn one programming language as a result of fulfilling a credit requirement

Next Steps

- Another round of teacher interviews is currently being conducted
- We have developed and started conducting student cognitive interviews which involve solving tasks in Scratch and Java and then comparing the code presented side-by-side
- We hope to schedule classroom observations in the next phase of the project
- Ultimately, we intend to develop supports or scaffolds that can be incorporated into the curriculum that will support greater transfer of knowledge from one programming language to another

*…**almost any of the structures like for loops**, and while loops, and conditionals, **they're again, syntactically different, but they're similar enough,** where when we say, if blah, then they make that connection very quickly, right? **They're like, "Oh, yeah. Okay, that's just an if statement from Python."***

*We talk abouts searching and sorting algorithms. Again, **we do it all in Python, but then when we do it in Java**, then it's not something that's brand new. **They see that the algorithm is still the same, it's just the implementation that differs.***

*I can bring it up to them and say like, "**Oh, remember how you did it last year**. We kind of did this with Tracy the Turtle where you move forward. This is how you're doing it in JavaScript, moving the car or whatever."*

***CodeHS is the thing that I use** with my kids nowadays*

Subcategory	Code	Description
Spontaneous Transfer	spontaneous	Teacher expresses seeing students transfer knowledge of one language to another of their own accord
Productive Transfer Resources	productive resources	Teacher mentions materials that purposely help transfer coding knowledge between languages
Curricular Support of Transfer	activities	Teacher mentions specific activities that support transfer
	curriculum structure	Teacher mentions aspects of the curriculum that support transfer
	teacher connections	Teacher makes explicit connections between multiple languages in their curriculum
Teacher Supports	explicit	Teacher includes explicit comparisons between multiple languages in their teaching
	pacing	Teacher mention where and when they draw explicit connections between multiple languages
	lessons	Teacher mentions specific lessons where language comparisons are productive
	materials	Teacher provides examples of materials or resources they use to support transfer

*But I think explicitly pointing out like, okay, you see how this is, and then you see how this is over here. You see the relationship and **having that explicit conversation, I think just solidifies their understanding***

*…**we're going to revisit some of the projects we did in Scratch, in p5.js**, and have them recreate a few of those things*

***I like Code.org where it did have that structure** has it where it can help guide in future ones.*

*But it is **at the beginning of each of those topics**, maybe, is where I initially mention it*

***For loops in Python look significantly different from a standard for loop in Java.** But once you point out that, you remind them that, "Well, range is just a generator and it's generating values from zero up to whatever, and we're taking those values and putting them in for i. Now, let's look at the Java version where we say, i starts at zero, and making that almost drawing arrows between the two. See how they're related?"*

This work is supported by the National Science Foundation through the EHR Core Research in STEM Learning and Learning Environments program (Award #2201209).

WestEd

Jennifer Houchins, Rosalind Owen, Bryan Matlen, and Yvonne Kao

Are you interested in participating in research? Join our mailing list →