The Role of Spatial Orientation in Diagram Design for Computational Thinking Development in K-8 Teachers

Jean Salac
salac@uw.edu
University of Washington
Seattle, WA, USA

Donna Eatinger
dmeatinger@uchicago.edu
University of Chicago
Chicago, IL, USA

Diana Franklin
dmfranklin@uchicago.edu
University of Chicago
Chicago, IL, USA

ABSTRACT

The worldwide push for computing education at younger ages requires that teachers are prepared to deliver instruction that supports all learners. Other discipline-based education research fields offer a wealth of instructional scaffolds worthy of exploration in computing. One such scaffold drawn from math education is diagramming. While diagrams are frequently employed in university computing, little is known about its applications in K-8 (ages 6-14) computing.

To inform diagram design for K-8 computing, we investigated how the spatial orientation of a diagram (horizontal or vertical) influenced the extent to which K-8 teachers developed different technological, pedagogical, and content knowledge (TPACK) of computational thinking (CT) concepts, such as loops, conditionals, and decomposition. We found that more teachers were able to decompose a sequence of events when using a vertical diagram. While teachers in both conditions were similarly able to describe various CT concepts and aspects of TPACK, more teachers using a vertical diagram made connections between concepts, whereas more teachers using a horizontal diagram described concepts in isolation. We hope this exploration will spur future work into diagramming and more broadly, spatial reasoning in K-8 computing.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; *Computational thinking*; *K-12 education.*

KEYWORDS

diagrams, teacher professional development, computational thinking, TPACK, K-8, spatial reasoning

ACM Reference Format:
Jean Salac, Donna Eatinger, and Diana Franklin. 2023. The Role of Spatial Orientation in Diagram Design for Computational Thinking Development in K-8 Teachers. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569737

1 INTRODUCTION

With the global growth of computing education at younger ages [13], it is critical that teachers are equipped to provide computing instruction that supports all learners. One way to support all learners is to provide them with sufficient scaffolding as they are introduced to new concepts. Other discipline-based education research fields, such as math [5, 23, 34], reading [4, 35] and science [23], are rich with instructional scaffolds that can potentially be adapted for computing. Diagramming is one such strategy. Although diagrams are a staple in university computing [17], little is known about its use in primary and secondary computing — how they should be designed, which concepts are they appropriate for, how they support both students and teachers, among other avenues for exploration.

To inform diagram design for primary and secondary computing, we investigated two different orientations of a diagram, vertical and horizontal, in a virtual professional development for teachers of students ages 10-14. In particular, we focus on the following research questions:

(1) How does diagram orientation influence the learning of decomposition in teachers?
(2) How does diagram orientation influence the development of technological pedagogical content knowledge (TPACK) in teachers?

We found that more teachers were able to partially or fully decompose a sequence of events when using a vertical diagram, as opposed to a horizontal diagram. Our results also revealed that teachers in both conditions were similar in their project completion rates, and their coverage of computational concepts and aspects of TPACK. However, more teachers who used a vertical diagram made connections between CT concepts, while more teachers who used a horizontal diagram described concepts in isolation. The *depth* of their understanding differed, suggesting connections to emerging work on spatial reasoning in computing [20, 21, 24–27].

2 BACKGROUND

2.1 Matrix Taxonomy: Bloom's for Computing

Fuller et al. proposed the Matrix Taxonomy [9], a two-dimensional adaptation of Bloom's taxonomy [2, 15] for computing. The two dimensions are 'Interpreting', the ability to understand an existing product, and 'Producing', the ability to design and build a product. The levels in the 'Producing' dimension are *Apply* and *Create*. In our study, we ask teachers to Use (explore the existing code), Modify (change the existing code), and Create (develop their own code) [16]. This enables teachers to demonstrate their ability to produce code, with the Use–>Modify task at the *Apply* level and the Create task at the highest 'Producing' level.

2.2 SOLO Taxonomy

To ground our teachers' understanding, we use the Structure of Observed Learning Outcomes (SOLO) taxonomy. It arranges learning

SIGCSE '23, March 15–18, 2023, Toronto, ON, Canada.

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569737

outcomes by structural complexity [1]. The SOLO taxonomy is comprised of five hierarchical levels of understanding: (1) prestructural, where nothing is known about the subject or task, (2) unistructural, where only one relevant aspect is known, (3) multistructural, where several relevant independent aspects are known, (4) relational, where aspects of knowledge are integrated into a structure, and (5) extended abstract, where knowledge is generalized into a new domain.

Scholars have employed the SOLO taxonomy to classify student responses. For example, Lister et al [19] compared responses to "explain in plain English questions" from students and educators. They found that most educators demonstrated extended abstract responses, while most students only demonstrated relational and multistructural responses. Similarly, we used the SOLO taxonomy to categorize teacher interview responses.

2.3 TPACK Model

TPACK is a framework for teacher knowledge for technology integration [14]. It describes the interactions of three critical bodies of knowledge: content, pedagogy, and technology. Pedagogical content knowledge (PCK) is the knowledge teachers have about their content and the best ways to teach it [11]. Technological pedagogical knowledge (TPK) is the knowledge teachers have of how teaching and learning can change when technology is used in particular ways [14]. Technological content knowledge (TCK) is the knowledge teachers have of how technology and content influence and constrain each other [14]. Lastly, TPACK is the knowledge teachers have of the representation of concepts using technology, the pedagogical techniques that use technology in constructive ways to teach content, and the difficulties students face when learning and how technology can help redress them [14].

The TPACK model has been used by scholars to analyze the different funds of knowledge of K-12 CS teachers. For example, Giannakos et al. used TPACK to examine the abilities and needs of a national sample of teachers in upper secondary education [10]. Vivian and Falkner utilized TPACK to study teachers' contributions to an online teacher professional development [36]. In this study, we also leveraged the TPACK model to characterize the knowledge expressed by teachers in their interviews.

2.4 Learning Strategies

Numerous visual computing learning strategies, such as tracing [18] and diagramming [6, 17] have been found to be effective at the university level [7]. However, strategies in the university setting are unlikely to directly translate in the K-8 setting. Students at that age may not be able to self-regulate their learning as well as university students, and also learn computing in vastly different contexts from university students.

Other discipline-based education research (DBER) fields provide insights into diagramming learning strategies for the primary and secondary setting. Timelines [4] and Venn diagrams [35] scaffold the organizing of key concepts and the understanding of larger themes. The "Draw-It" Problem Solving strategy supports students with learning disabilities in solving word problems in math through diagramming [34]. Concept maps and vee diagrams [5, 23] support metalearning and problem-solving skills in math and science.

By comparison, prior work in learning strategies in K-8 computing, let alone diagramming strategies, is thin. Examples of K-8 learning strategies include Use–>Modify–>Create [16], PRIMM [33], and TIPP&SEE [32]. The first provides more scaffolded, guided instruction for each concept, followed by a more open-ended task to encourage student agency and creativity. PRIMM guides teachers in creating scaffolded activities in text-based programming languages to encourage learning, while TIPP&SEE guides students through previewing and deliberate tinkering of Scratch projects. While all of these strategies have been shown to be effective, they are largely procedural or text-based scaffolds.

This work extends prior research by investigating how the design of diagrams, drawn from work in university computing and other DBERs, can support the development of computational thinking and TPACK in K-8 teachers, with the ultimate goal of practicing this strategy with their students. In particular, we focused on spatial orientation in diagram design, with promising new work suggesting links between spatial reasoning and computing. Margulieux proposed Spatial Encoding Strategy Theory to explain the relationship between spatial skill and STEM achievement [21]. At the university level, spatial skills has been linked to computer science success [20, 25–27]. At the K-8 level, Parker et al. found that students struggled more with code matching questions where an animal moved in a grid vertically compared with horizontally [24].

3 METHODS

3.1 Study Context

This study took place with teachers from all over the United States, where computing teachers frequently start as teachers of other subjects and receive professional development to teach computing. In Summer 2020, 45 teachers who taught students ages 10-14 participated in a virtual professional development (PD) for Scratch Encore, an intermediate Scratch curriculum designed for students with a year of introductory programming experience [8]. The curriculum is comprised of modules covering computational thinking concepts such as repetition, conditionals, and decomposition [28–30].

The PD covered one module per week, with a 30 minute synchronous introduction, an hour-long coding session in groups of 4-6, and teachers working through the materials asynchronously between the two sessions. All materials were adapted for virtual instruction using Google forms and slides.

Each module follows a Use–>Modify–>Create structure, where a concept is introduced using example code before open-ended exploration [16]. In Scratch Encore, the Use–> Modify step is further scaffolded with the TIPP&SEE learning strategy [32]. Finally, they are prompted to create an open-ended artifact incorporating the new concept.

In this study, we focus on a module called *Decomposition by Sequence*. It was designed based on the Decomposition learning trajectory [28] and targeted Scratch programming language-specific constructs. In particular, this module focuses on a sequence of events across multiple sprites, where different actions were triggered by between-sprite interactions for which Scratch provides sensing blocks (touching color or touching sprite). The learning goals for this module are for students to learn to (1) decompose a sequence of events into separate actions and their triggering events, (2) create scripts that will trigger the action of one sprite dependent

on the action of another sprite, (3) use sensing blocks to stop and start actions, and (4) plan and create an animation based on a set of events and actions.

In this module, we designed horizontal and vertical versions of a diagram to scaffold the learning of these goals. As prior experience can influence how much scaffolding they would need, teachers were assigned to either the horizontal (H) or vertical (V) condition such that each condition would have the same median years of computer science teaching experience. 22 teachers were in the horizontal condition, with a median of 4 years of experience. 23 teachers were in the vertical condition, with a median of 4.25 years.

To fill out the diagram in the "Use/Modify" worksheet for this module, teachers first watched a video that demonstrated a complete Scratch project with a sequence of events. Based on the video, they completed a partially-filled out diagram that decomposes the sequence of events. After they filled out the diagram, the worksheet directed them to an incomplete version of the demonstrated Scratch project, which they explored using the TIPP&SEE learning strategy. Using their completed diagrams to support their coding, teachers were then prompted to modify the project such that it worked like the demonstrated project. In the "Create" worksheet, teachers were prompted to plan their culminating projects with the help of an empty fill-in-the-blank diagram (Figures 1 & 2).

3.2 Diagram Design

In this module, the objective of the diagram was to scaffold the decomposition of the conditional interactions between multiple sprites. From prior iterations of Scratch Encore, teachers reported struggles with previous diagrams. Based on teacher feedback, classroom observations, and prior student work, we revised the diagram, with existing CS diagrams (e.g. control flow diagrams) serving as a subject matter guide and diagramming strategies from math education serving as a pedagogical guide [34]. This diagram underwent several rounds of revision, with feedback from researchers and practitioners at each round.

For this study, we designed two orientations of the revised diagram — horizontal and vertical. The horizontal version, which is read from left to right, is similar to the way students would read natural language text in English and aligns with typical representations of timelines [4]. In contrast, the vertical version, which is read from top to bottom, is akin to the way programs are read and aligns with the code structure of Scratch.

Figures 1 and 2 depict examples of the vertical and horizontal "Create" diagrams, respectively. The blue boxes represent student input. In this example, a student wants to program a sequence of events from a soccer game where a player runs and kicks a ball into a goal with spectator cheers. The events in this interaction are when the player touches the ball and when the ball touches the goal. The player's actions are to run until they touch the ball. The ball's actions are to stay still until the player touches it, after which it rolls until it touches the goal. The goal's actions are to stay still until the ball touches, after which it plays a celebratory sound. The scripts for the player, ball, and the goal are shown in Figure 3.

3.3 Data Analysis

3.3.1 Diagram Worksheets. We analyzed teachers' diagrams in the "Create" worksheet for evidence of decomposition, where they

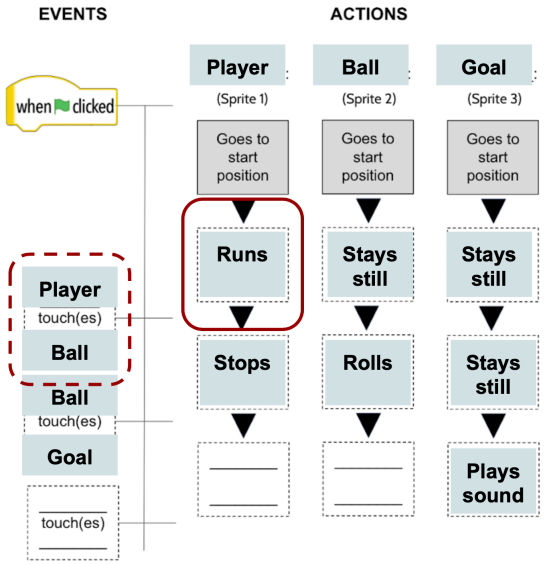

Figure 1: Vertical Diagram with Complete Decomposition

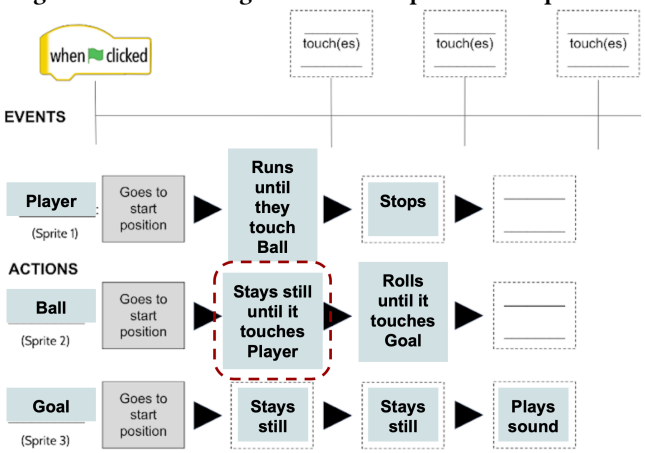

Figure 2: Horizontal Diagram with No Decomposition

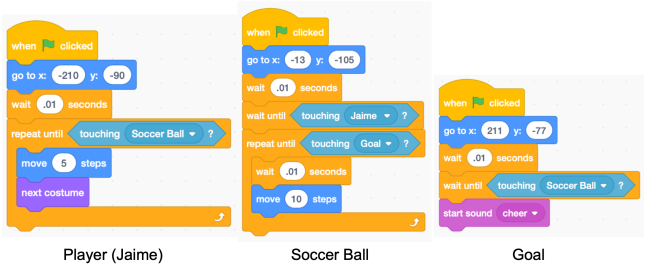

Figure 3: Code for Each Example Sprite

planned their own final project. The first two authors developed a coding manual of categories that described the extent to which they decomposed the sequence of events — separated all the events and actions (complete decomposition), separated at least one event or action (partial decomposition), or did not separate events and actions (no decomposition). Figure 1 depicts a complete decomposition, with the events (e.g. "when Player touches Ball") to the left of

the timeline and actions (e.g. "Runs") to the right. Figure 2 depicts no decomposition with the events and actions together below the timeline (e.g. "Ball stays still until it touches the Player").

The manual also classified characteristics of responses in the "event" and "action" boxes in the diagram. In Figure 1, the event boxes are the boxes to the left of the vertical timeline while the action boxes are to the right. In Figure 2, the event boxes are the boxes above the horizontal timeline while the action boxes are below. Correct responses would be writing one event in an event box (e.g. box in the dotted red circle from Figure 1) and one action in an action box (e.g. box in the solid red circle from Figure 1). Common incorrect responses included writing in an action box: an event (e.g. box in the dotted red circle from Figure 2), a sprite (e.g. if they wrote "Player" in that box), and multiple actions (e.g. if they wrote "stays still then rolls" in that box).

The first two authors coded each worksheet separately and then resolved any disagreements through discussion following the consensus-based model from Hammer and Berland [12]. To see if there was a dependence between condition and the extent of decomposition, we used a Chi-square test of independence; we report χ and p values. This test, however, was not appropriate for their response characteristics as the expected value for each cell was less than five [22].

3.3.2 Scratch Projects. Features of teachers' culminating "Create" Scratch projects were automatically scraped to determine if they fulfilled the assigned requirements. There were five requirements: adding a new backdrop, using at least 3 sprites, using the `go to x: y:` block in at least 2 sprites (for initialization), animating at least 2 sprites, and using a `repeat until` or `wait until` blocks to program a sequence of events. Additionally, there were two extra extensions that teachers could complete: adding a new event to a third sprite and using a sound block. Lastly, we also checked if teachers used `broadcast/receive` blocks as it was a frequently mentioned theme in interviews. To see if there was a statistically-significant difference between the two conditions' completion rates, we conducted the Chi-squared test on proportions, from which we provide χ and p values.

3.3.3 Semi-structured Interviews. At the end of the PD (1 month after the module), retrospective semi-structured interviews were conducted with some teachers to learn more about their mental models of their final Scratch projects and to better understand to what extent diagrams scaffolded their learning. 13 teachers in each group (horizontal vs vertical) were interviewed for about 15-20 minutes and were selected based on availability. Teachers were able to reference their completed diagrams and Scratch projects throughout the interview. The interview protocol is in Table 1.

Interview transcripts were first open coded by the first two authors to identify emerging themes. Based on these themes, a qualitative coding manual was developed, covering CT concepts, the levels in the SOLO taxonomy, and aspects of the TPACK framework (section 2). The two authors coded each interview separately and then resolved any disagreements through discussion. The Chi-square test of independence was not appropriate to use on the themes identified in the interviews as the expected value for each cell was less than five, with some cells even being zero [22].

Topic	Questions
Warm Up	Can you show me what your project does? Was this diagram helpful to you in planning your project? What went well? What was confusing? Do you have any suggestions for improvement?
Diagram	Can you explain your thinking as you filled out this diagram? How did you fill it out for *Sprite X*? How did you decide what to write in this box? What went well? What was confusing? Can you explain how you used this diagram while coding your Scratch project? How did you program *Sprite X*? How did you program *event/action X*? What went well? What was confusing?
TPACK	How would you explain how your project works to your students? How would you explain how to fill out this diagram to your students? Do you think this diagram would be helpful for your students? Do you have any suggestions for student-facing diagrams?

Table 1: Teacher Interview Protocol

4 RESULTS

We first present results from analyzing diagram worksheets and end-of-module projects, followed with results from teacher interviews.

Finding 1: 78% of teachers in the vertical condition decomposed the sequence of events in their diagrams, either completely or partially, compared with 56% of teachers in the horizontal condition.
Figure 4 depicts the proportion of teachers who did not separate events and actions, separated at least one event or action, and separated all events and actions when decomposing a sequence of events. A majority of teachers in the vertical condition either completely or partially decomposed the sequence of events into its constituent events and actions, while a majority of the teachers in the horizontal condition did not decompose or only partially decomposed the sequence of events. The dependence between condition and the extent of decomposition, however, was not statistically significant ($\chi = 4.08, p = .129$).

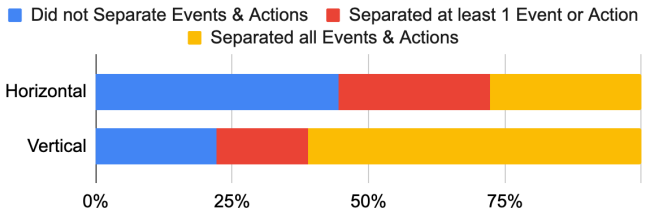

Figure 4: Diagram Worksheet Features

Taking a closer look at the worksheets, we found that more vertical teachers described one action in each action box (10 V vs 4 H) and one event in each event box (12 V vs 8 H). This demonstrates

a complete decomposition of the sequence of events. In contrast, when filling out the action boxes, more teachers in the horizontal condition wrote at least one event or sprite, more than one sequential action, and actions that did not belong to the sprite in that row/column (Figure 5). This suggests partial or no decomposition of the sequence of events. The mistakes that were more common among the vertical teachers were idiosyncratic responses in the action boxes, such as x-y coordinates or time, and writing at least one sprite in an event box. The latter mistake could be interpreted as incomplete events, as events in this module involve conditional sprite interactions.

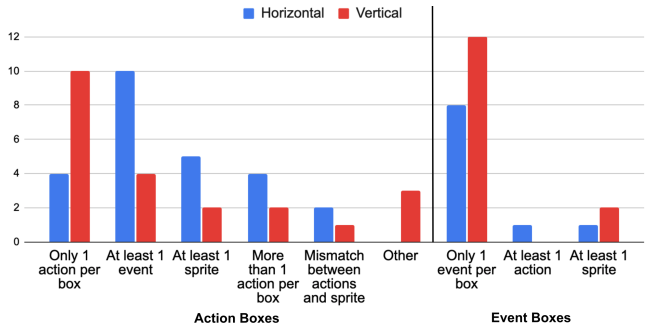

Figure 5: Diagram Worksheet Responses

Finding 2: Teachers in both conditions completed project requirements at similar rates.

Figure 6 shows the completion rates of project requirements and extensions (adding third sprite with a new event and adding a sound block). Chi-square tests on proportions revealed no statistically-significant differences in the completion rates of any of the requirements or extensions. The only exception is the sound block extension, which vertical teachers were more likely to complete ($\chi = 3.91, p = .0479$). In addition to the stated project requirements and extensions, we also analyzed the usage of `broadcast/receive` blocks in their projects based on the information gathered during teacher interviews. These blocks are commonly used for message passing and synchronization in Scratch, and had not been introduced in this module or prior lessons. While the difference was not statistically significant, it is important to note that more horizontal teachers used these blocks to program their final project, instead of using the new `repeat until` or `wait until` blocks introduced in this module.

Finding 3: When explaining their projects or diagrams, teachers in both conditions were similar in the CT concepts they described.

Figure 7 details the number of teachers in each condition who described each CT concept when prompted to explain their Scratch projects or diagrams. While teachers in both conditions were similar in the CT concepts they detailed, it is interesting to note that nearly twice as many teachers in the vertical condition brought up *decomposition*, the focal CT concept in this module.

Finding 4: Teachers in both conditions were similar in the different types of knowledge in the TPACK model they expressed in interviews.

Figure 8 illustrates the number of teachers in each condition who exhibited each aspect of TPACK. The two singular aspects of TPACK, technological and content knowledge, were omitted from the graph; the former because none of the teachers described technological knowledge alone and the latter because all of the teachers exhibited some level of content knowledge. A similar number of teachers in both conditions exhibited different aspects of TPACK (Figure 8).

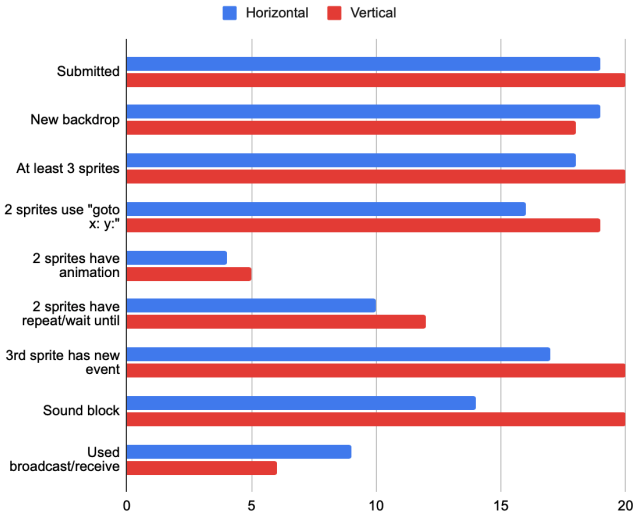

Figure 6: End-of-Module Project Features

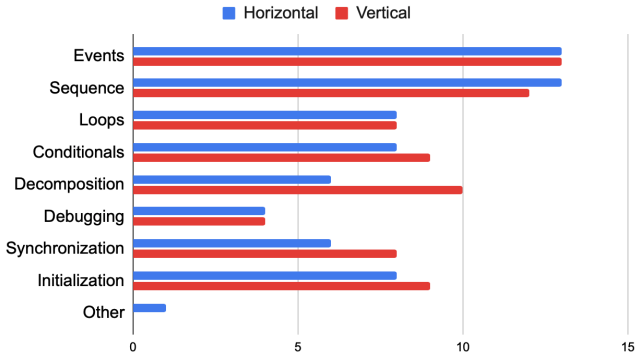

Figure 7: CT Concepts Described in Interviews

The most commonly expressed types of knowledge were pedagogical knowledge (PK) and pedagogical content knowledge (PCK). A teacher in our study exhibited their PK when describing how they would pair struggling students with more advanced students for peer instruction, while another teacher displayed their PCK when drawing students' attention to different events in a thinkaloud explanation of their project. One teacher exhibited technology content knowledge, who explained that sprites in Scratch need a `goto x: y:` block to initialize in the correct starting position. Six teachers demonstrated technological pedagogical knowledge, such as a teacher explaining XY coordinates before students learned them in math so they can move their sprites in Scratch. Lastly, five teachers even exhibited TPACK, demonstrating an intersection of technological, pedagogical, and content knowledge. For example, one teacher described making videos of them modeling their project, breaking down the different events and actions, so that their students could watch their explanations at their own pace.

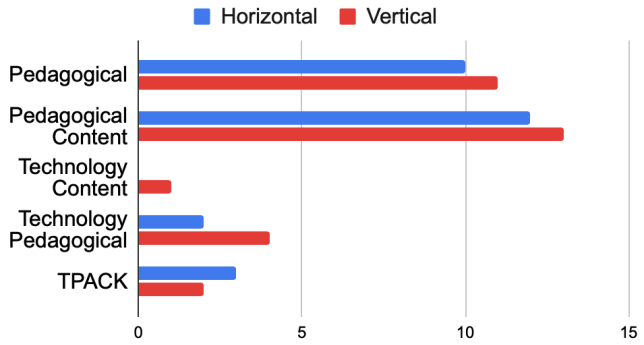

Figure 8: TPACK Aspects Described in Interviews

Finding 5: More teachers in the vertical condition described CT concepts at the relational level (11 V vs 4 H) of the SOLO taxonomy.

Figure 9 displays the number of teachers who demonstrated an understanding of CT concepts at each level of the SOLO taxonomy (Section 2). The lowest level in the hierarchy, the prestructural level, was omitted as all teachers knew at least one relevant concept. A unistructural explanation consists of one relevant concept. Both multistructural and relational explanations are comprised of several concepts, but connections between concepts are made in relational explanations.

Most teachers in the horizontal condition explained CT concepts at the unistructural and multistructural levels, while the explanations of most teachers in the vertical condition were at the multistructural and relational levels. An example of a unistructural explanation would be a teacher in the horizontal condition who described the conditional interaction in their project as "The first person is saying that she needed to get the blue potion[...]And then the elf lady says something else and then a little fairy drags over and tries to get her potion." While their project consisted of blocks corresponding to other concepts, they only described sequence. For a multistructural example, a teacher in the horizontal condition said, "When she touches the guitar, it would start playing[...]then it would send out a message to the target and it would repeat until the color red was touching that the brown of the boy's shoes". They described the concepts of events, sequence, loops, and synchronization, but the concepts were specific to their project and not integrated into a broader structure. In contrast, a relational example would be from a teacher in the vertical condition: "Sprites are characters that you can program to interact with each other in many different ways...when a sprite reaches that position and comes in contact with another sprite,[...]there's going to be an action that occurs when two sprites come in contact with each other." While this teacher described fewer concepts than the teacher with the multistructural explanation (only events and sequence), this teacher linked the concepts into a larger structure.

5 DISCUSSION

For RQ1, our analysis of diagram worksheets revealed that most teachers using the vertical diagrams either partially or completely decomposed the sequence of events into their composite sprite actions and events. In the horizontal condition, most teachers either

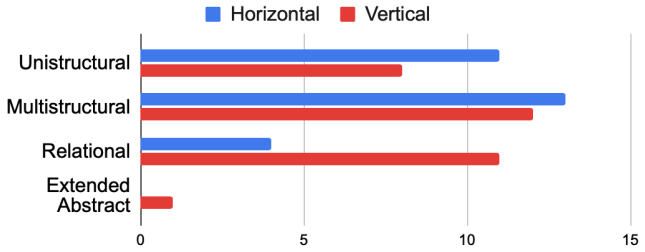

Figure 9: SOLO Taxonomy Levels of Interview Responses

partially or did not decompose the sequence of events. While teachers in each condition differed in the degree of decomposition in their diagrams, they performed similarly in their Scratch projects, completing requirements at similar rates. This result suggests that, regardless of condition, teachers were able to produce code artifacts at the highest 'Producing' level of the Matrix Taxonomy, *Create* [9]. However, prior work has shown that students frequently create with code that they do not fully understand [3, 31]. A similar phenomenon may have occurred with the teachers in the horizontal condition, given their responses to the diagram worksheets and their unistructural and multistructural descriptions of CT concepts in interviews based on the SOLO taxonomy.

For RQ2, we found that teachers in both conditions developed some level of content knowledge, one of the pillars of TPACK. They described CT concepts with similar frequency when asked to explain their projects or diagrams. While we could not run statistics on the CT concepts described, it is important to note that nearly double the number of teachers in the vertical condition mentioned *decomposition*, the key concept in this module. Teachers in both conditions also exhibited other aspects of TPACK with similar frequency, with the two most common aspects of TPACK being pedagogical and pedagogical content knowledge. Nonetheless, most teachers in the vertical condition demonstrated deeper knowledge of the CT concepts, either describing multiple concepts (multistructural) or drawing connections between several concepts (relational). In contrast, most teachers in the horizontal condition described concepts in isolation (unistructural or multistructural).

While our small sample size limited the suitability of quantitative analysis, our results, combined with our qualitative analysis, provided insights into diagram design for the learning of computational thinking concepts, such as decomposition and conditionals. The reason for performance differences across different spatial orientations may be further motivation for nascent research on spatial reasoning in computing [20, 21, 24–27]. Most interestingly, our results with teachers exhibit a reverse pattern of Parker et al's results with students, who found horizontal easier than vertical movement in coding questions [24]. This may be because of the different study populations, nature of the task (code writing in our study vs reading in [24]), the spatiality of language [37], or something else entirely; further research would be needed to uncover the underlying reasons. We hope that this study will lead to further research into diagrams, and spatial reasoning more broadly, in K-8 computing.

ACKNOWLEDGMENTS

This project was funded by National Science Foundation (NSF) Grant No. 1738758 and DGE-1746045.

REFERENCES

[1] John B Biggs and Kevin F Collis. 1982. *Evaluation the quality of learning: the SOLO taxonomy (structure of the observed learning outcome).* Academic Press.

[2] Benjamin S Bloom et al. 1956. Taxonomy of educational objectives. Vol. 1: Cognitive domain. *New York: McKay* (1956), 20–24.

[3] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, Vol. 1. 25.

[4] Kristy A Brugar and Kathryn Roberts. 2014. Timelines: An opportunity for meeting standards through textbook reading. *The Social Studies* 105, 5 (2014), 230–236.

[5] AJ Cañas, P Reiska, M Åhlberg, and JD Novak. 2008. Concept mapping & vee diagramming a primary mathematics sub-topic:"Time". (2008).

[6] Kathryn Cunningham, Sarah Blanchard, Barbara Ericson, and Mark Guzdial. 2017. Using Tracing and Sketching to Solve Programming Problems: Replicating and Extending an Analysis of What Students Draw. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 164–172. https://doi.org/10.1145/3105726.3106190

[7] Katrina Falkner, Rebecca Vivian, and Nickolas JG Falkner. 2014. Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 291–296.

[8] Diana Franklin, David Weintrop, Jennifer Palmer, Merijke Coenraad, Melissa Cobian, Kristan Beck, Andrew Rasmussen, Sue Krause, Max White, Marco Anaya, et al. 2020. Scratch Encore: The design and pilot of a culturally-relevant intermediate Scratch curriculum. In *Proceedings of the 51st ACM technical symposium on computer science education*. 794–800.

[9] Ursula Fuller, Colin G Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L Lewis, Donna McGee Thompson, Charles Riedesel, et al. 2007. Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin* 39, 4 (2007), 152–170.

[10] Michail N Giannakos, Spyros Doukakis, Ilias O Pappas, Nikos Adamopoulos, and Panagiota Giannopoulou. 2015. Investigating teachers' confidence on technological pedagogical and content knowledge: an initial validation of TPACK scales in K-12 computing education context. *Journal of Computers in Education* 2, 1 (2015), 43–59.

[11] Sigrun Gudmundsdottir and Lee Shulman. 1987. Pedagogical content knowledge in social studies. *Scandinavian Journal of Educationl Research* 31, 2 (1987), 59–70.

[12] David Hammer and Leema K Berland. 2014. Confusing claims for data: A critique of common practices for presenting qualitative research on learning. *Journal of the Learning Sciences* 23, 1 (2014), 37–46.

[13] Peter Hubwieser, Michail N Giannakos, Marc Berges, Torsten Brinda, Ira Diethelm, Johannes Magenheim, Yogendra Pal, Jana Jackova, and Egle Jasute. 2015. A global snapshot of computer science education in K-12 schools. In *Proceedings of the 2015 ITiCSE on working group reports*. ACM, 65–83.

[14] Matthew Koehler and Punya Mishra. 2009. What is technological pedagogical content knowledge (TPACK)? *Contemporary issues in technology and teacher education* 9, 1 (2009), 60–70.

[15] David R Krathwohl. 2002. A revision of Bloom's taxonomy: An overview. *Theory into practice* 41, 4 (2002), 212–218.

[16] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *Acm Inroads* 2, 1 (2011), 32–37.

[17] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. 2004. A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 119–150.

[18] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *Acm sigcse bulletin* 41, 3 (2009), 161–165.

[19] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin* 38, 3 (2006), 118–122.

[20] Anna Ly, Jack Parkinson, Quintin Cutts, Michael Liut, and Andrew Petersen. 2021. Spatial Skills and Demographic Factors in CS1. In *21st Koli Calling International Conference on Computing Education Research*. 1–10.

[21] Lauren E Margulieux. 2020. Spatial encoding strategy theory: The relationship between spatial skill and stem achievement. *ACM Inroads* 11, 1 (2020), 65–75.

[22] Mary L McHugh. 2013. The chi-square test of independence. *Biochemia medica* 23, 2 (2013), 143–149.

[23] Peter Akinsola Okebukola. 1992. Attitude of teachers towards concept mapping and vee diagramming as metalearning tools in science and mathematics. *Educational Research* 34, 3 (1992), 201–213.

[24] Miranda C Parker, Leiny Garcia, Yvonne S Kao, Diana Franklin, Susan Krause, and Mark Warschauer. 2022. A Pair of ACES: An Analysis of Isomorphic Questions on an Elementary Computing Assessment. In *Proceedings of the 2022 ACM Conference on International Computing Education Research V. 1*. 2–14.

[25] Miranda C Parker, Amber Solomon, Brianna Pritchett, David A Illingworth, Lauren E Marguilieux, and Mark Guzdial. 2018. Socioeconomic status and computer science achievement: Spatial ability as a mediating variable in a novel model of understanding. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 97–105.

[26] Jack Parkinson and Quintin Cutts. 2019. Chairs' AWARD: investigating the relationship between spatial skills and computer science. *ACM Inroads* 10, 1 (2019), 64–73.

[27] Jack Parkinson and Quintin Cutts. 2020. The effect of a spatial skills training course in introductory computing. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 439–445.

[28] Kathryn M Rich, T Andrew Binkowski, Carla Strickland, and Diana Franklin. 2018. Decomposition: A k-8 computational thinking learning trajectory. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 124–132.

[29] Kathryn M Rich, Diana Franklin, Carla Strickland, Andy Isaacs, and Donna Eatinger. 2020. A Learning Trajectory for Variables Based in Computational Thinking Literature: Using Levels of Thinking to Develop Instruction. *Computer Science Education* (2020), 1–22.

[30] Kathryn M Rich, Carla Strickland, T Andrew Binkowski, Cheryl Moran, and Diana Franklin. 2018. K–8 learning trajectories derived from research literature: sequence, repetition, conditionals. *ACM Inroads* 9, 1 (2018), 46–55.

[31] Jean Salac and Diana Franklin. 2020. If they build it, will they understand it? Exploring the relationship between student code and performance. In *Proceedings of the 2020 ACM conference on innovation and technology in computer science education*. 473–479.

[32] Jean Salac, Cathy Thomas, Chloe Butler, Ashley Sanchez, and Diana Franklin. 2020. TIPP&SEE: A Learning Strategy to Guide Students through Use - Modify Scratch Activities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 79–85. https://doi.org/10.1145/3328778.3366821

[33] Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teachers' Experiences of using PRIMM to Teach Programming in School. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 476–482.

[34] Delinda van Garderen and Amy M Scheuermann. 2015. Diagramming word problems: A strategic approach for instruction. *Intervention in School and Clinic* 50, 5 (2015), 282–290.

[35] Sharon Vaughn and Meaghan Edmonds. 2006. Reading comprehension for older readers. *Intervention in school and clinic* 41, 3 (2006), 131–137.

[36] Rebecca Vivian and Katrina Falkner. 2019. Identifying Teachers' Technological Pedagogical Content Knowledge for Computer Science in the Primary Years. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 147–155.

[37] Benjamin Lee Whorf. 2012. *Language, thought, and reality: Selected writings of Benjamin Lee Whorf*. MIT press.