Journal of Computer Assisted Learning

WILEY

Journal of Computer Assisted Learning

ORIGINAL ARTICLE

Scaffolding Coding Instruction Through Literacy via the Compose and Code Digital Platform and Curriculum

Amy Hutchison[1] | Qi Si[2] | Jamie Colwell[3] | Erdogan Kaya[4] | Eileen Jakeway[4] | Brittany Miller[4] | Kristie Gutierrez[3] | Kelly Regan[4] | Anna Evmenova[4]

[1]Department of Curriculum and Instruction, The University of Alabama, Tuscaloosa, Alabama, USA | [2]School of Teacher Education and Leadership, Utah State University, Logan, Utah, USA | [3]Department of Teaching and Learning, Old Dominion University, Norfolk, Virginia, USA | [4]College of Education & Human Development, George Mason University, Fairfax, Virginia, USA

Correspondence: Amy Hutchison (achutchison1@ua.edu)

Received: 30 July 2024 | **Revised:** 12 December 2024 | **Accepted:** 21 December 2024

Funding: This work was supported by US National Science Foundation (Grant number 1837380).

Keywords: coding | computer science education | digital learning | writing

ABSTRACT

Background: In recent years, computer science education has emerged as a necessary part of school curricula for students of all ages. With such momentum in this direction, it is essential that program designers, educators, and researchers ensure that computer science education is designed to be inclusive, effective, and engaging for all students.

Objective: Accordingly, this paper reports on the design and implementation of an inclusive digital learning platform and accompanying curriculum for scaffolding and integrating coding into writing instruction for elementary-aged students (approximately ages 9–12). In this paper, we report on teachers' uses of the Compose and Code (CoCo) platform and curriculum, how students used its features, and its influence on students' computational thinking skills and attitudes about coding.

Method: Data analysed in this mixed-methods study come from 11 teachers and 595 students in Grades 3–6. Data sources included teacher reflections and interviews, an assessment of computational thinking for students, and a coding attitudes survey for students. Quantitative data were analysed descriptively and using paired sample *t*-tests. Qualitative data were analysed inductively using open coding to determine emergent categories.

Results and Conclusion: Findings indicate that (1) a majority of students effectively used the CoCo platform to plan their work and code in Scratch, with a smaller percentage using the self-evaluation and self-monitoring features, (2) teachers indicated overall positive perceptions of the CoCo platform and curriculum, with strong support for using it in the future, (3) students' computational thinking skills improved over the course of the project, with results indicating a large effect size ($g = 1.24$), and (4) student attitudinal results were mixed, providing insights to the barriers that students face when learning to code. Overall, this study indicates that the CoCo platform and curriculum show promise as a scaffolded, structured, and integrated tool for teaching elementary computer science to elementary grade students.

1 | Introduction

In recent years, computer science education has emerged as a necessary part of school curricula for students of all ages, with a concerted effort to integrate computer science into elementary school curricula (Vegas, Hansen, and Fowler 2021). This unprecedented expansion of computer science education at the elementary level is underpinned by several factors, one of which is the ever-increasing demand for a tech-savvy workforce (Piliouras et al. 2014). Though, there are numerous potential

Summary

- What is currently known about this topic?
  - Visual programming languages such as Scratch can be effective tools for teaching elementary grade students how to code.
  - There is some evidence that this type of instruction needs to be carefully scaffolded in order to make it accessible to all learners.
- What does this paper add?
  - This paper reports on student and teacher perceptions of an inclusive digital learning platform and accompanying curriculum for scaffolding and integrating coding into elementary writing instruction and its effect on computational thinking skills and student attitudes.
  - To our knowledge, the Compose and Code curriculum is the only existing approach that strategically leverages the similarities between writing and coding to use traditional writing as a bridge for learning about coding.
- Implications for practice
  - Results show that the use of the Compose and Code digital graphic organiser tool and curriculum that integrates writing and coding (a) helped scaffold the process of coding a story in Scratch into small, manageable steps and made it possible for all students to participate in coding instruction, (b) reduced distraction by helping students pre-plan *what* they wanted to code so that they would not be diverted by irrelevant options and features, and (c) led to improved computational thinking skills, which often results in a flashy animation with a fuzzy message.
  - These findings indicate the importance of strategic integration of learning support tools and strategies when teaching coding to children.

benefits of learning about computer science that expand beyond future career opportunities. First, coding, a key aspect of computer science, can be a creative process that enables students to express themselves in ways they otherwise could not. Even young students can create interactive stories and games, using programming languages to translate their ideas into a digital product (Bers 2020). Not only does coding enable students to creatively express themselves, but scholars have found that coding can foster creativity among young children (Murcia et al. 2020; Su et al. 2024). Further, coding requires computational thinking skills and the breakdown of complex problems into smaller, manageable steps, all of which are crucial skills applicable far beyond the computer science realm (Li et al. 2020). Additionally, research has shown that computing experiences can improve students' attitudes about STEM more broadly (e.g., Master et al. 2017). In other words, the integration of computer science into elementary education is not merely a response to workforce demands but a crucial step in fostering creativity, problem-solving skills, and positive STEM attitudes in young learners.

Despite the potential benefits and increasing necessity of teaching computer science to young students, it can be difficult to add content to an already-crowded instructional day. As evidence of this difficulty, elementary teachers repeatedly report that lack of instructional time is a primary reason for not integrating computer science and coding into instruction and report that integrating computer science into existing subject area content is ideal (Colwell et al. 2023) Further, it is difficult to ensure that all students have the opportunity to learn computer science since it is often as an advanced learning option rather than an inclusive option that is available to students with a range of learning abilities (Hutchison et al. 2021). These challenges prompted a novel approach. To address these obstacles, the current study leveraged the similarities between writing and coding to create an inclusive, scaffolded computer science curriculum and digital learning platform for integration into elementary literacy instruction. The current study was designed to guide students in using traditional writing as a bridge for learning about coding. This approach was selected because of the similarities between the logical structures of writing and coding, which have been pointed out by Vee (2017) and others. Both writing and coding demand planning, sequencing, decomposition of complex tasks, and adherence to specific syntax (grammar versus code syntax). This parallel may allow students to apply familiar writing skills to understand coding structure and logic, for example, applying narrative structure comprehension to grasp program sequencing. The iterative process of writing—editing and revising—mirrors debugging in coding. Furthermore, the emphasis on planning and outlining in writing translates directly to the need for organised code, leading to more efficient and accurate programming. Additionally, many scholars have argued that coding is an increasingly essential form of communication (e.g., Vee 2017; Hutchison, Nadolny, and Estapa 2016; Hutchison et al. 2021), making it appropriate for inclusion in literacy instruction, the content area in which elementary students are primarily taught about methods of communication.

By integrating coding within existing writing instruction, rather than adding extra time, this approach maximises efficiency and may increase student interest through multimodal learning (combining familiar writing with novel coding). This synergistic approach may also make coding skills more relevant, enhancing both writing instruction and computer science education. Accordingly, this paper reports on the design and implementation of the Compose and Code digital learning platform and accompanying curriculum for scaffolding and integrating coding into elementary writing instruction. The current study addresses the need for inclusive approaches for integrating computer science into elementary instruction, specifically by leveraging the existing time and skills developed during elementary writing instruction. In this paper, we report on teachers' uses of the Compose and Code (CoCo) platform and curriculum, how students used its features, and its influence on students' computational thinking skills and attitudes about coding.

2 | Literature Review

Scholars worldwide have designed and studied a variety of curricula and programs to teach computer science, coding and computational thinking in the early years (e.g., Bers et al. 2014; Sullivan and Bers 2016). Computational thinking (CT) has been defined and studied many ways; for purposes of the current

study, we focused on CT skills broadly, using programming as an approach for equipping students with the CT skills of sequencing, pattern recognition, abstraction, decomposition, and algorithmic thinking (Shute, Sun, and Asbell-Clarke 2017). Examples of how scholars have previously studied CT and coding include: (1) developing CT learning games to determine how game play may impact CT skills (e.g., Asbell-Clarke et al. 2021; Ayman et al. 2018); (2) considering how robots and physical computing can help teach CT skills for young children (e.g., Casey et al. 2018; Egbert et al. 2021; Katterfeldt et al. 2018); (3) developing culturally responsive computing curricula (e.g., Novak and Khan 2022); and (4) considering how to integrate computer science concepts into other elementary content areas (e.g., Hutchison et al. 2021; Deniz et al. 2020; Yesilyurt et al. 2022). Even with this increase in efforts to understand approaches for teaching computer science in the elementary grades, there is still a significant need to consider and understand approaches for students who have difficulty learning these concepts. Examples of approaches for supporting students of all levels include application of self-regulated strategy development (Sanders et al. 2019) and the use of comprehension tools such as graphic organisers, both of which were used in the development of the CoCo platform and lessons. Each of these approaches is described subsequently.

2.1 | Graphic Organisers

Graphic organisers are visual tools that aid in organising and representing information, concepts, and relationships in a structured and easily understandable format. Graphic organisers can act as comprehension tools to help students structure their thoughts and understand the logical steps involved in coding (Hutchison et al. 2024). They can be used to reduce cognitive load and aid in planning before actually writing code (Lee and Tan 2010). Various types and uses of graphic organisers have been shown to increase comprehension of many kinds of content. For instance, research by Mayer and Moreno (2003) indicates that these tools can reduce cognitive load during multimedia learning. Graphic organisers can assist learners in organising information into manageable segments, facilitating a better understanding of relationships between ideas, concepts, and facts. For example, DiCecco and Gleason (2002) found that students with disabilities benefitted from the use of a graphic organiser to improve understanding of the relationships among concepts derived from expository text. Additionally, when used for writing, computer-based graphic organisers may improve both the quantity and quality of writing (Evmenova et al. 2016). Further, computer-based graphic organisers are effective for students both with and without disabilities (Sturm and Rankin-Erickson 2002; Ponce, Mayer, and Lopez 2013). Based on the aforementioned research, we reasoned that a graphic organiser format may be particularly helpful for students who have difficulty with the abstract nature of programming. Thus, the CoCo platform used in the current study was designed as a type of graphic organiser.

2.2 | Self-Regulated Strategy Development

Self-regulated strategy development (SRSD) is a research-based instructional approach designed to improve students' writing skills, particularly for individuals with writing difficulties or learning disabilities. This approach places a strong emphasis on explicitly teaching academic strategies and self-regulation skills, and there is substantial evidence supporting its effectiveness for learning (Sanders et al. 2019). The cultivation of self-regulation skills empowers students to set learning goals and monitor their progress, thereby increasing the likelihood of task completion. Specifically, SRSD is comprised of six instructional stages: (a) developing and activating background knowledge, (b) discussing the strategy, (c) modelling the strategy, (d) memorising the strategy, (e) supporting students in using the strategy, and (f) promoting independent performance (Sanders et al. 2019). Numerous studies have demonstrated that SRSD can improve the writing skills of students with and without disabilities and highlight the positive impact of explicit strategy instruction (e.g., Kistner et al. 2010). In the context of coding, we reasoned that SRSD may help students break down complex tasks, plan their code, and debug effectively. This strategic approach to self-regulation is important because coding requires significant self-direction and problem-solving skills. Accordingly, SRSD was included as a key design feature of the CoCo platform and lessons used in the current study.

In sum, the CoCo platform employs a digital graphic organiser format to help students plan their code, utilises SRSD to empower self-regulated learning, and combines coding with writing to improve comprehension and expression. The research questions then evaluate the effectiveness of this integrated approach on student CT skills and attitudes.

2.3 | The Current Study

The current study fills the need for an inclusive, integrated approach for teaching computer science to elementary students and examines the use of the Compose and Code platform for scaffolding and integrating writing and coding. The study addresses the following research questions (RQs):

RQ1. To what extent do students use the features of the Compose and Code platform for scaffolding and integrating writing and coding?

RQ2. What are teachers' perceptions of the Compose and Code platform and curriculum?

RQ3. To what extent does participation in Compose and Code instruction influence students' computational thinking skills?

RQ4. To what extent does participation in Compose and Code instruction influence students' attitudes about computer science?

3 | Methodology

3.1 | Research Design

As part of a larger study, this work was conducted using a mixed-methods approach to understand teacher and student perceptions and outcomes related to utilising the CoCo platform

and its accompanying curriculum. Compose and Code (CoCo) is an inclusive digital learning platform designed to support all students in developing computational thinking and coding skills through explanatory writing. Functioning as an interactive graphic organiser, CoCo incorporates self-regulated learning strategies and Universal Design for Learning supports (including a graphic organiser, audio comments, video models, and text-to-speech) to scaffold the process of coding animated stories using the Scratch platform. One of its unique features is that it guides students in planning out a written sequence of ideas and then generating a plan for how to use their written text as a foundation for coding an animated story in Scratch. Once students select what they want to do in Scratch, CoCo automatically populates the relevant Scratch code blocks, providing visual cues and video tutorials on code implementation. This feature is important for students who need a more scaffolded approach for learning abstract coding concepts. CoCo guides students in computational thinking, prompting them to decompose their big idea into small steps and plan a logical sequence for their code. Five levels of scaffolding ensure accessibility for students with varying writing and coding abilities and diverse learning needs. CoCo guides students through sequential steps and self-monitoring throughout the writing and coding process. For example (as illustrated in Figure 1), a student might use CoCo to plan a how-to guide for making hot chocolate, decomposing the process into steps and then visually mapping out the corresponding code blocks needed to create an animated project in Scratch.

Prior to using CoCo and the accompanying curriculum, teachers participated in online professional development to learn about foundational concepts related to the work and to learn about the digital tools they would use. The topics of the online professional development modules were: (1) What is computer science and why is it important?; (2) What is computational thinking and how is it relevant to you and your students?; (3) How do I code and how can I fuse computational thinking into plugged activities?; (4) What is universal design for learning and how do I use it?; and (5) What is CoCo?

Following participation in the online professional development, teachers implemented four researcher-designed units of study focused on composing written explanatory compositions and then planning and translating those compositions into multimedia animations using Scratch visual block-based programming environment. Lessons were implemented during the literacy block of the regular school day and implementation occurred over a period of 4 months, with one unit implemented each month. Researchers observed participating teachers at least one time each and reflected with teachers about the lesson implementation and provided support for future implementation. Additionally, teachers submitted a reflection to the researchers after each lesson and researchers responded to questions, concerns, and outcomes. The units included scripted lesson plans and ready-made slide decks that were designed to be used with CoCo. Further, to make the lessons inclusive of all learners, all lessons were designed with the Universal Design for Learning framework and components of SRSD (Sanders et al. 2019). The researchers used a checklist for each lesson to ensure that each lesson included the following components related to SRSD: (1) provides explicit instruction when building new knowledge; (2) engages students through active participation, frequent eliciting of student responses, and a brisk pace;

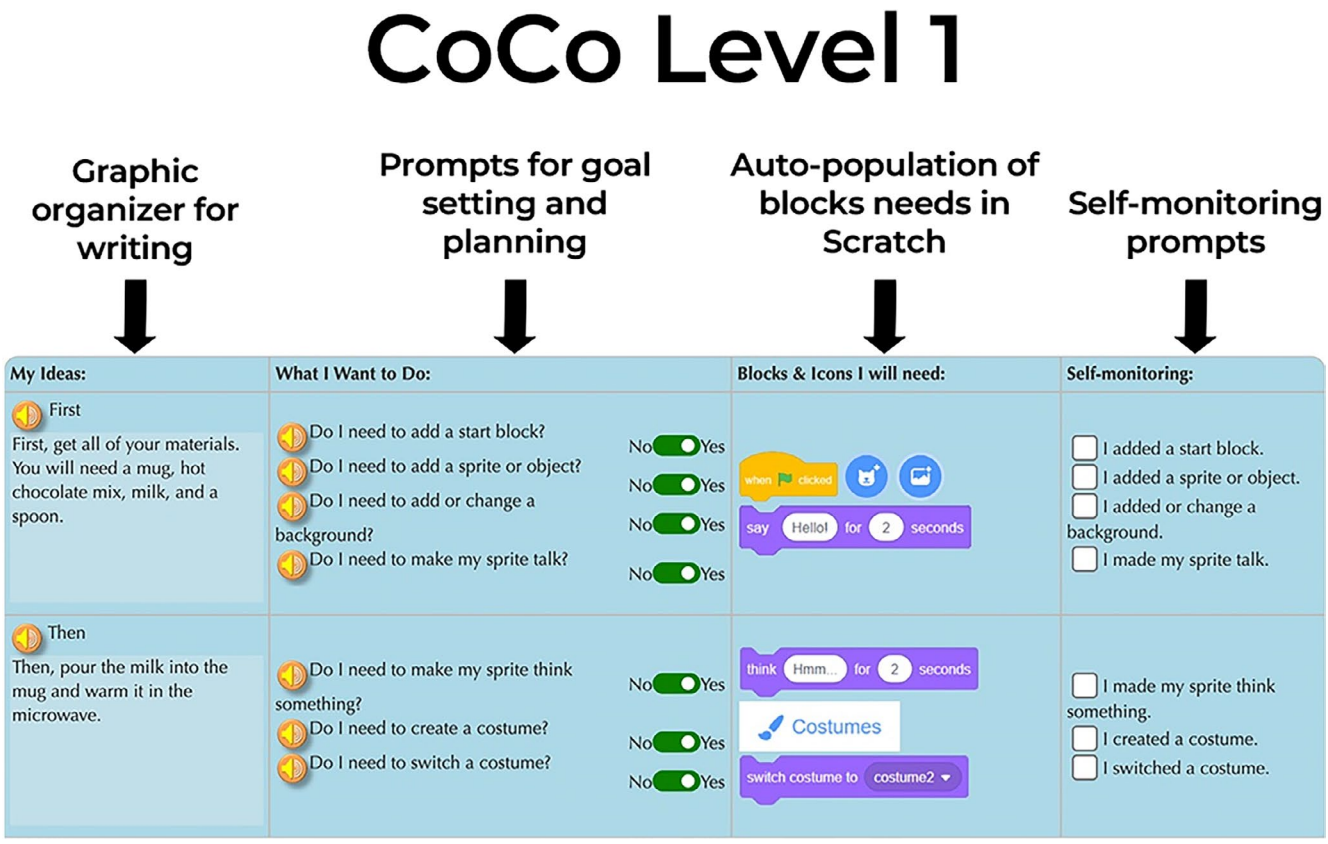

FIGURE 1 | Features of the compose and code platform.

(3) includes strategies to scaffold and meet student needs (e.g., assistive technology, graphic organisers, self-regulation learning strategies, and peer-assisted learning); (4) provides a clear structure to the lesson with a limited number of measurable learning objectives and opportunities to activate and recall prior knowledge; (5) incorporates universal design for learning principles; multiple means of engagement, multiple means of representation, and multiple means of action and expression; and (6) motivates students with varied methods of instruction, by identifying the purpose of learning objectives and frequently revisiting objectives to show accomplishments, and providing clear and specific praise and reinforcement.

The CoCo lessons are grouped into four units, each including explicit instruction on computational thinking skill(s), learning objectives in Scratch, and a particular text type. In all cases, students use CoCo as the graphic organiser for planning both their coding and their writing. In each unit, students are introduced to new Scratch blocks both in the direct instruction and in the options provided by the levels of CoCo. Table 1 shows the foci of each unit.

To provide sufficient scaffolding for classes to complete the units at a reasonable pace, each lesson was designed to gradually introduce additional Scratch blocks and features of CoCo. The lessons build on each other such that students' progress from less complex tasks like brainstorming or unscrambling an existing animation in Scratch to independently writing and coding their own compositions and animations within the unit. Units range from three to five lessons, and the research team also devised a condensed five-lesson sequence that introduces students to the key ideas of computer science in lesson 1, to Scratch and CoCo in lesson 2, to explanatory writing in lesson 3, to planning an animation in lesson 4, and then finishing and sharing their multimedia animations in lesson 5. Two different versions of the lessons were created, one for third and fourth grade and one for fifth and sixth grade. The distinction between the two versions is that the higher grade level materials include more complex programming concepts and text types.

TABLE 1 | Description of instructional units.

Unit	Foci
Unit 1 5 lessons	CS or CT focus: pattern recognition, sequencing, + numerous Scratch blocks & coding concepts Writing focus: writing a recipe
Unit 2 4 lessons	CS or CT focus: abstraction, + numerous Scratch blocks & coding concepts Writing focus: Writing an expository composition of the students' choice, that is, instructions for how to play a game, directions to a location, and so forth, using the transition words 'First, Then, Next, and Last' as organisational anchors
Unit 3 3 lessons	CS & Writing integrated: Using knowledge about the text genre, Scratch blocks, and CoCo structure from previous units, students use Scratch and CoCo together to create and engaging multimedia animation based on the explanatory text they generated in CoCo
Unit 4 3 lessons	CS or CT focus: decomposition + numerous Scratch blocks & coding concept Writing focus: Students write story summaries and apply new CS concepts to code animated versions of their summaries

3.2 | Participants

Teacher participants in this study included 11 teachers (*Female* = 9, *Male* = 2) from 6 different schools in a diverse school district in a Mid-Atlantic state in the United States. Teachers' years of teaching experience ranged from 4 to 15 years. Additionally, there were 595 student participants in grades 3–6, grade level distributions as follows: grade 3 $n = 203$, grade 4 $n = 101$, grade 5 $n = 210$, grade 6 $n = 81$. School administrators did not allow us to collect demographic information on individual students. However, the school district in which the study took place serves a diverse student population of more than 181,000 students from grades pre-kindergarten through 12th grade. Specifically, over 27% of the total students are from economically disadvantaged families; about 14% of the students are reported as students with disabilities; over 20% of the students are identified as multilingual learners. Demographically, about 37% of the students are White, 27% are Hispanic, 20% are Asian, 10% are Black, and 6% of the students are from other ethnic groups. Although we were unable to collect information about individual students' disability status, lessons took place in inclusive classrooms in which general education and special education teachers taught together, indicating that a portion of the students had documented disabilities and individualised education plans.

3.3 | Data Sources and Data Analysis

To answer the first research question, we examined the compositions that students created in the CoCo platform throughout the course of the project. We calculated the frequency with which students used each feature of the platform as percentages (see Figure 1 for features of the CoCo platform). Additionally, we analysed field notes from classroom observations to add further detail about students' uses of CoCo. Field notes were first read holistically, with researchers highlighting any notes that specifically referred to how students used and interacted with the CoCo platform. This led to a total of 17 relevant data excerpts, which researchers then compared to look for patterns in the data. Something was considered to be a pattern if it appeared in observation notes three or more times. This analysis process resulted in the identification of three patterns related to students' use of CoCo, which we report in Section 4.1.

Figure 2 shows an example of a student's planning for writing using the CoCo platform and the resulting Scratch project based on the writing and planning done in CoCo.

Several data sources were used to answer the second research question. First, teachers completed brief reflections after teaching each unit, for a total of 4 reflections per teacher. These reflections asked teachers to indicate (1) their level of preparedness to teach the instructional unit, (2) how well the

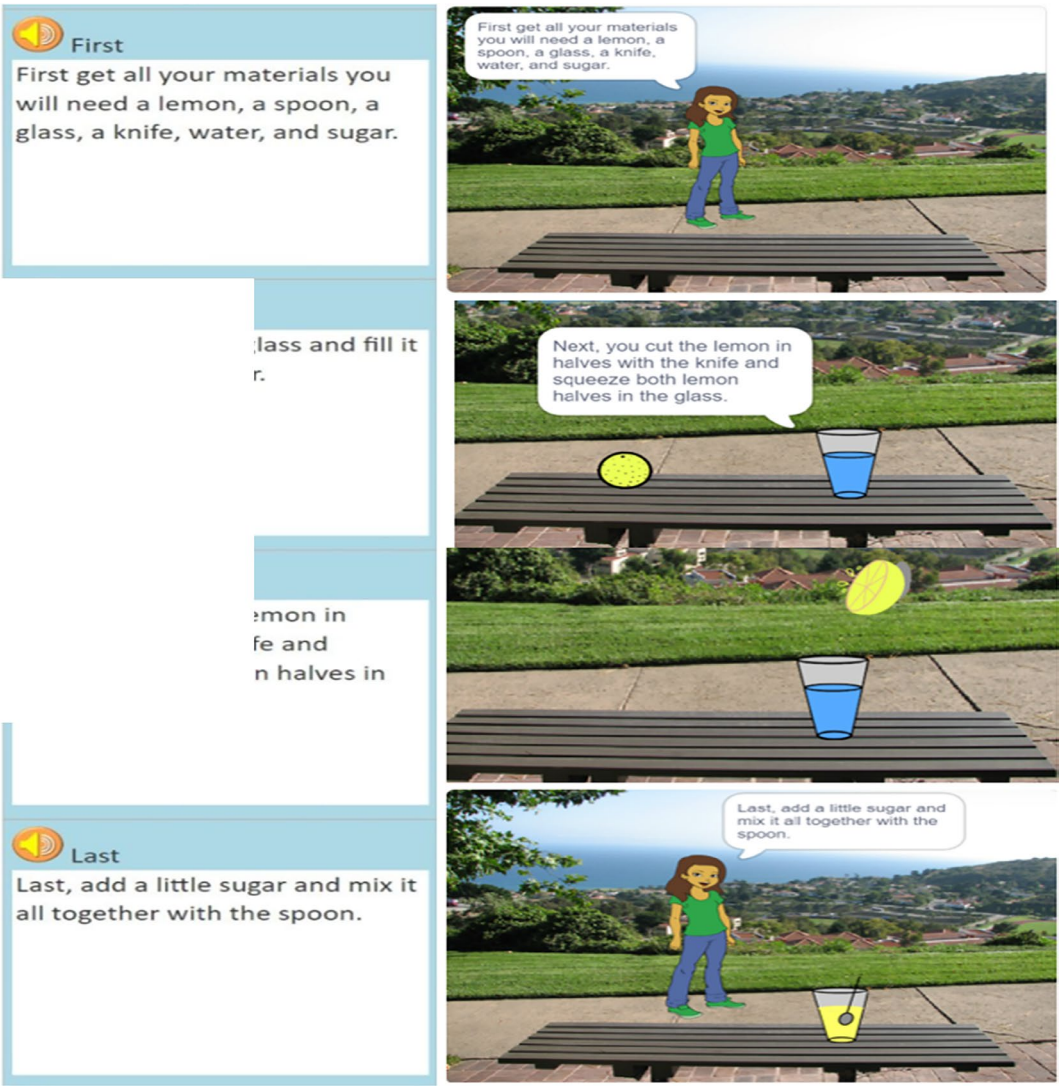

FIGURE 2 | Example of student work planned in CoCo and then coded in scratch.

TABLE 2 | Example of using data to determine category trends.

Data excerpt	Codes	Category trend
'Along with that, the **graphic organiser on CoCo was helpful**. It **forced students to have a plan** instead of just jumping right in to creating. It helped them **slow down their thinking**'. (3rd Grade Teacher)	CoCo as helpful CoCo as useful for planning CoCo as scaffold	Positive teacher perception of CoCo for supporting students in coding

CoCo platform and instructional materials worked for students with and without disabilities, (3) an overall rating of the unit and the most positive and negative aspects of using the platform and other instructional materials, (4) what could be done to make the instructional materials better, and (5) whether they would implement the unit again with or without changes. Additionally, teachers completed a post-participation interview in which they responded to a series of semi-structured questions regarding their perceptions of participating in the project. The interviews were transcribed and analysed using open coding following a general inductive approach (Thomas 2006). Semi-structured interview data that focused on teachers' perceptions of the CoCo platform and curriculum were first transcribed and read through holistically to gain an overall understanding of the data and emerging ideas. Transcribed data were then inductively coded to determine emergent category trends related to teacher perceptions. Table 2 illustrates this process:

To answer the third research question, students completed the Elementary Student Computing Attitudes Survey ([ESCAS]; Mason and Rich 2020) at the beginning of the Fall 2022 semester and the end of the Spring 2023 semester. The survey instrument comprises 23 items assessed using a 6-point Likert-type

scale which includes the following options: (1) strongly disagree, (2) disagree, (3) somewhat disagree, (4) somewhat agree, (5) agree, and (6) strongly agree (refer to Appendix A). The ESCAS was developed and validated through a rigorous process involving over 6000 upper-elementary students. Mason and Rich (2020) provided evidence for content validity through expert review and field testing. They provided evidence for construct validity using confirmatory factor analysis (CFA) with good model fit (RMSEA and SRMR < 0.08, CFI and TLI > 0.9), and structural equation modelling (SEM), also showing good fit (RMSEA = 0.037, CFI = 0.950, TLI = 0.940, SRMR = 0.032). To score the responses, we assigned a score of 1 to 6 for each of the response categories on the scale from strongly disagree to strongly agree. We explored the Cronbach's alpha reliability of the 23 items with all the pre-survey responses ($n = 557$), and the alpha value of 0.91 indicates a relatively high reliability for the whole instrument. The present study aimed to compare the potential differences between pre-survey and post-survey data, so we only included paired data from both pre-and post-survey tests for a total of 238 completed responses.

To answer the fourth research question, students' computational thinking skills were measured at the beginning of the Fall 2022 semester and the end of the Spring 2023 semester using the *Assessment of Computing for Elementary Students* (ACES; Parker et al. 2021) as pre- and post-tests (see Appendix B for a copy of the assessment). Parker et al. (2021) conducted a systematic development process for the ACES and collected evidence for its validity, which included a pilot study with 57 elementary students. The researchers gathered evidence for content validity through multiple methods, including conducting cognitive interviews with students, aligning the assessment with established computational thinking learning trajectories, and basing the assessment on existing curricula and assessments for similar age groups. They provided evidence for construct validity using item analysis, with acceptable difficulty indices (mostly between 0.2 and 0.8) and discrimination indices (mostly above 0.2). Evidence for reliability was provided via Cronbach's alpha ($\alpha = 0.686$).

The ACES included 10 multiple-choice questions to measure elementary students' computational thinking skills. To score the responses, we assigned a score of 1 to each question, so the maximum score for this assessment is 10. Among the 10 questions, five questions have only one correct answer while the other five questions have more than one correct answer. For the questions that have more than one correct answer, we assigned scores proportionately to the percentage of the response that was correct. A total of 237 students completed both pre- and post-assessments.

4 | Results

4.1 | RQ1: To What Extent Do Students Use the Features of the Compose and Code Platform for Scaffolding and Integrating Writing and Coding?

The main purpose of the CoCo platform is to help students plan out the content that they want to code before going to Scratch to create their project. CoCo guides students in creating written content and deciding how to animate their story in Scratch to most effectively convey their main idea(s). Once students check the boxes to indicate what they want to include in their Scratch animation, the correct block appears in CoCo to help students see which blocks they will need to find in Scratch. Once done planning in CoCo, students go to Scratch to code their animated story. To understand how often students use the features of CoCo and how it may have supported them in coding their animated story, researchers designed a rubric with the following four sections: (1) relevance of the final Scratch project to the prompt provided in CoCo; (2) use of each feature of the CoCo graphic organiser; (3) connection of the Scratch project to the written story planned in CoCo; and (4) extent to which the animations added in Scratch support the main ideas and overall purpose of the story. Using this rubric, researchers scored, and students planned work in CoCo with the final projects created in Scratch; Unit 1 ($n = 336$), Unit 2 ($n = 96$), Unit 3 ($n = 273$), Unit 4 ($n = 286$).

Results indicate that a large percentage of students effectively used the CoCo platform to plan their written product and to plan how they would code their product in Scratch. Across the four units, an average of 71% of students planned out their written content and used the transition words provided in the first column of CoCo graphic organiser. In addition, across units, an average of 65% of students planned what they wanted their Scratch project to look like and the message they wanted it to convey using the second column of the CoCo organiser. A review of Scratch projects across all 4 units indicated that 70% of the projects created in Scratch closely aligned with the content planned in CoCo. The least used features of the CoCo platform were the self-monitoring questions and self-evaluation features, both of which are SRSD strategies intended to keep students focused on their goals. The self-monitoring feature provided a column where students could return to CoCo to check off whether they had found the correct blocks in Scratch. The self-evaluation feature asked students to indicate how they felt about the written and coded portion of their project. Only about 31% of the students used the self-monitoring and self-evaluation features in CoCo.

Field notes from classroom observations provided further insight into students' uses of CoCo, revealing three relevant patterns in how CoCo was used or implemented. First, across classrooms, researchers noted that teachers often did not explain or encourage students to use the self-monitoring and self-reflection features in CoCo. In all cases, this was due to time constraints. Self-monitoring was explained as one of the last steps in all the lessons, encouraging students to continue returning to CoCo as they coded in Scratch. However, during most observations, it was noted that students often had just enough time to complete their coding, with little to no time to self-monitor to see if they had included all of the blocks and ideas they had planned in CoCo. Self-reflection was designed as one of the final steps of the lessons to help students reflect on their work as a whole. However, researchers never observed teachers getting to this point in the lessons because they all ran out of time before getting to this part. This lack of time likely explains why these were the least used features of the CoCo platform.

The second pattern that researchers identified is that teachers often asked students to write their ideas on paper *before*

beginning to use the CoCo platform. In classrooms where this was the case, researchers noted that students began their writing in CoCo quickly and were more prepared to enter information into Column 1 of CoCo. In these classrooms, students often got further along in the lesson than they did in classrooms where students did not first do some initial brainstorming or planning on paper. Planning on paper was one of the options suggested in the lesson plans, but was listed as optional.

A third pattern that was identified during observations was that students enjoyed and made use of the text-to-speech feature that enabled them to hear what they had written in Column 1 of CoCo read aloud to them. We do not have data about the exact number of times that students accessed this feature, but researchers noted that students often identified mistakes in their work or revised their work after using this feature.

Although teachers were not specifically asked about which features students used most frequently, interview data from teachers also provides some insight into how students used CoCo. For example, in their interviews, over half of the participating teachers ($n = 6$) indicated that they required students to use CoCo before coding in Scratch and monitored them to ensure they had completed their planning in CoCo. This monitoring that occurred before coding may be indicative of why the pre-planning features were used more than the self-monitoring features. A sixth-grade teacher explained why she was diligent in monitoring their planning:

> I mean it's like any graphic organizer, they just want to get going and it's good for them to pause and think about it. Coco was actually helpful to them in organizing because a lot of them wanted to just start coding, and [by using CoCo] they knew what they wanted to do when they got to Scratch.

A majority of teachers ($n = 6$) also commented that the first column of CoCo, in which students plan out their story, was beneficial. Teachers indicated that it required students to write and think, even when they didn't want to, and that the step-by-step process helped them to plan successfully. For example, a third-grade teacher stated that column one helped students

> ...break down their thinking, to take it step by step and really just think about the elements of literature; Who their characters are, what is the setting, the plot. So it helped them to really focus on each individual element.

4.2 | RQ2: What Are Teachers' Perceptions of the CoCo Platform and Curriculum?

To understand the teachers' perceptions of the usefulness of the CoCo platform and curriculum, teachers were asked to complete a reflection after teaching each of the four instructional units. Each reflection consisted of 10 multiple-choice questions about teachers' levels of preparedness to teach the lessons and use CoCo, the extent to which the lessons and CoCo platform helped meet the needs of diverse students, and open-ended questions about general impressions of the lesson materials and CoCo platform.

Across the four instructional units, 80% of teachers felt mostly or fully prepared to facilitate the units. 90% of teachers reported that the units mostly or fully meet the needs of students who are on-grade-level or above-grade-level for reading and writing, whereas 62% and 60% of teachers, respectively, reported that the units mostly or fully met the needs of students reading and writing below grade level and learning English as a second or other language. 40% of teachers indicated that they encountered unexpected surprises while teaching the units, which resulted in teachers reporting that they chunked the units due to time constraints (53%), students' abilities (21%), and students' language background (7%). About 84% of the teachers indicated that they would implement the instructional units again, indicating high usability and acceptability from teachers.

Results from interviews supported reflection results. Teachers indicated positive overall perceptions of the CoCo platform and curriculum, with some considerations for minor barriers they faced. These results are described in the following two sub-sections.

4.2.1 | CoCo Platform

Teachers indicated appreciation for the scaffolded approach that the CoCo platform provided students to plan for coding. For example, the fifth-grade teacher noted, 'I thought that the CoCo platform for the students was really great. I think giving them that level of just planning, especially for this project where some of them were coming in with very minimal knowledge of how to do it, I think was quite helpful'. Teachers liked that CoCo was manageable for students, automatically generated blocks to help students visualise what they would need to select and use to build code in Scratch, and made students take time to plan out their ideas, much in the way students are encouraged to map out and plan traditional writing.

Teachers commented, too, on how the CoCo platform supported their students learning to speak English as a Second Language (ESL) and their students with learning disabilities. A third-grade teacher reflected,

> It's been interesting seeing the range of which students [CoCo] works for better than others. I think I like the concept a lot. I like how it forces kids to really sit and plan, and I love that using CoCo just shows them the blocks of what they would need to do, and I like that it has that self-monitoring tool.

Teachers regularly commented on the usefulness of CoCo as a planning tool and support for students who were unfamiliar with Scratch or who needed additional support for language or learning needs. Additionally, interview results noted that CoCo forced students to consider multiple aspects of using Scratch and coding

a story or how-to script online before students ever entered the Scratch platform. They felt that this level of preparation created a more seamless Scratch experience and final work product.

One teacher provided an overview of why she thought each feature of CoCo was useful, commenting as follows:

> I think it was useful because it broke it down for the kids step by step for their writing. I liked the audio component because they could play back their writing. And then sometimes if they hear it... You can tell kids, 'Reread, reread, reread, check for mistakes'. But hearing it, they could realize, 'Oh, I did not type or write what I thought I was writing'. I liked the way with CoCo, they were in control of their choices by clicking yes or no, the way they had the visual pop-up of the commands. I also liked the way you could click... There was a place where you could click to remind yourself of what that block did. So I thought that was all very useful.

> I like that there is a checklist where they go back and check off, did I do this. For some kids, they would go back and forth. For some kids, they would get so caught up in their creating with Scratch that I would have to say, 'Oh, go back to your checklist'. They needed that constant reminder- have you done everything? And then at the end, the reflection piece, I think that's a very good piece for them to do. We always do some sort of reflection whenever we're doing some sort of work, so I thought that was a quick, positive way to do that. So I thought CoCo was very useful and very helpful.

The only specified drawback that was documented was students, particularly those with language or special learning needs, struggled to log into the CoCo platform as email and password information were unfamiliar to them.

4.2.2 | Curriculum

Like the CoCo platform, perceptions of the CoCo curriculum that was designed to accompany the CoCo platform and was provided to teachers were positive, with teachers highlighting multiple aspects of the curriculum as useful. First, teachers felt that the lesson plans were easy to follow, provided strong content, and provided numerous useful resources throughout. A third-grade teacher commented, 'The lesson plans were very helpful. They were thorough. They were descriptive, and it helped me every bit along the way. Even when there were days where I would need two days, it was because the lesson plans were very clear'. Other teachers commented that the lessons took more class time than the plans indicated. However, they felt that the pacing of the lesson, its presentation, and content were strong so that they could break lessons down to the specific timeframe they had to focus on computer science.

Teachers also commented on the benefit of the vocabulary specific to computer science that was included in the lesson plans and the scripts that accompanied the plans. Not only did teachers find value in repetitive exposure to specific terms associated with computer science and coding, but they also felt that this aspect of the lessons supported students' learning and understanding of coding and planning for coding. A teacher reflected,

> I found the scripted stuff very useful because there's a lot of heavy terminology in coding ... algorithm and stuff like that. You just forget those when you're talking to a bunch of third graders and intuitively, you're like, 'Oh, let me use an easier word'. But if it's written right there, then you can say both of them and be purposeful about repeating the words that they need to learn.

These types of comments indicated that the lessons reinforced repetition in vocabulary and that repetition was beneficial to students in their learning.

Finally, interview results indicated that, while teachers had differing perspectives on whether coding could be considered a literacy skill, teachers appreciated the added layer of understanding that coding and using the CoCo platform provided to students as they planned their final coded projects in Scratch. One teacher summarised this idea by describing her students' retelling of *The Three Little Pigs* story through Scratch:

> '[Scratch] wasn't just a story; it was something that they were physically doing. They had a part in it. They could understand. They could relate to it better. And it just seemed to be just an overall better experience for them other than just reading a story ... so that was really great'.

Such comments highlighted teachers' perceptions of the benefits of the multimodalities involved when students planned for and coded stories. Although interviews revealed minor considerations where project refinement may be needed, overall perceptions of the CoCo platform and the curriculum revealed high teacher satisfaction from participating in the project.

4.3 | RQ3: To What Extent Does Participation in Compose and Code Instruction Influence Students' Computational Thinking Skills?

A total of 237 students completed the ACES at both pre- and post-test. Several paired *t*-tests were conducted to explore the differences between pre-and post-assessments. As Table 3 presents, the results indicate statistically significant differences in students' scores from the pre-test ($M = 8.50$, $SD = 1.43$) to the post-test ($M = 8.94$, $SD = 1.22$), with the mean score increasing by 0.44 points on the post-test, $t\,(236) = 5.46$, $p < 0.001$. The results also indicate a large effect size (> 0.80) on post-assessment, Hedge's $g = 1.24$. When compared by grade level, results indicate statistically significant score increases from pre- to post-test for 3rd and 6th grade students. Specifically, 3rd-grade students gained 1 point from pre- to post-assessment ($p < 0.001$) with a large effect size ($g = 1.48$). There was an average score increase of 0.08 points from pre- to post-assessment for 5th grade students,

but the statistics indicate that the gain is non-significant ($p = 0.24$). Sixth grade students gained 0.35 points from pre- to post-assessment ($p < 0.001$) with a medium effect size ($g = 0.76$).

4.4 | RQ4: To What Extent Does Participation in Compose and Code Instruction Influence Students' Attitudes About Computer Science?

A total of 315 students from 3rd grade ($n = 161$), 5th grade ($n = 88$), and 6th grade ($n = 66$) completed both the pre- and post- Computing Attitudes Survey. Paired samples *t*-tests were conducted to compare differences in the overall mean score for the entire survey from pre-test to post-test. Results showed no significant difference in the overall score for the whole group between pre-survey ($M = 101.15$, $SD = 15.77$) and post-survey ($M = 99.71$, $SD = 17.73$). A one-way ANOVA was conducted to explore statistical differences among grade subgroups on the post-survey. The results indicate the differences between grade levels were statistically significant, $F\ (2, 312) = [3.89]$, $p < 0.05$. The partial eta-squared effect size 0.024 suggests a small effect size (> 0.01). The Tukey post hoc test indicated significant differences between 5th-grade ($M = 103.89$, $SD = 15.30$) and 6th-grade ($M = 96.38$, $SD = 17.46$), with 5th-grade students indicating higher levels of self-efficacy and interest than 6th-grade students by 7.51 points ($p < 0.05$) on post-survey.

Furthermore, more paired samples *t*-tests were conducted to compare differences in the mean score (M) for individual items on the survey. Table 4 compares the percentage results of the response categories for pre- and post-survey on each item. The asterisk in front of the items indicates the statistical significance ($p < 0.05$) of the mean score difference between pre- and post-survey. Specifically, 14 of 23 items indicate a significant difference.

5 | Discussion and Future Directions

The purpose of this study was to understand how students used the CoCo platform for computer science instruction, how teachers perceived the CoCo platform and accompanying lessons, and how students' computational thinking skills and computing attitudes changed as a result of participating in the lessons. As shown in the results, the CoCo platform and curriculum show promise as effective tools for teaching elementary computer science.

First, the results indicate that a large percentage of students made use of the features provided in CoCo, and a large majority of the products they created in Scratch closely aligned with what they planned in CoCo. This is an important finding because of what researchers have previously deemed the Flashy but Fuzzy effect (Hutchison et al. 2023). This effect refers to the phenomenon witnessed in previous research where students create flashy designs in Scratch, but with no clear meaning or message. CoCo was designed to guide students in planning their ideas *before* going to Scratch, with the hope that CoCo would: (a) scaffold the process of coding a story in Scratch into small, manageable steps, making it possible for all students to participate, and (b) reduce distraction by helping students preplan *what* they wanted to code so that they would not be diverted by irrelevant options and features, which often results in a flashy animation with a fuzzy message. Additionally, the CoCo lessons were carefully scaffolded to gradually introduce computer science and computational thinking concepts while gradually introducing new blocks and functions in Scratch. Collectively, the platform and lessons were intended to help students create code that successfully conveys their intended message. Our results show that students were largely able to do that. The least used features in CoCo were the self-monitoring and self-evaluation components. These self-regulated learning strategies were designed to help students stay on task and return to their pre-planned ideas when coding in Scratch, and to help them self-evaluate the outcome. It is plausible to consider that the alignment among students' planning in CoCo and the resulting Scratch projects may have been even greater if more students had used the self-monitoring and self-evaluation features, so, use of these features is something to explore in the future. As previously noted, teachers often ran out of time to explicitly guide students in using these features, so future research might focus on ways to help students continuously self-monitor and reflect throughout the entirety of the lessons. It is also important to consider that students may have monitored and reflected on their work in other ways that are not reflected in the CoCo platform. It may be beneficial for future research to include think-alouds with students as they work to determine the extent to which they are self-monitoring and reflecting to better understand the extent to which heavily scaffolded instruction enables self-monitoring and reflection.

Importantly, teachers reflected positively on the lessons, and a majority of teachers felt that the lessons met the needs of their students and would use the lessons again. This is an important finding for several reasons. First, although there are many emerging curricula focused on CS, to our knowledge, ours is the only one that integrates writing and CS in this way. This integration seems like a necessary evolution when considering the great extent to which people increasingly code websites and other media to convey their ideas or promote a topic of interest.

TABLE 3 | *t*-test Results for the ACES.

	N	Pre-test (*M* [*SD*])	Post-test (*M* [*SD*])	*t*	*df*	Sig. *p*	Hedge's *g* effect size
Total	237	8.50 (1.43)	8.94 (1.22)	5.46	236	< 0.001	1.24
Grade 3	77	7.41 (1.52)	8.40 (1.45)	5.91	72	< 0.001	1.48
Grade 5	99	9.05 (0.93)	9.13 (1.04)	0.71	98	0.24	—
Grade 6	60	9.00 (1.17)	9.35 (0.83)	3.56	59	< 0.001	0.76

TABLE 4 | Comparisons of options on survey items.

Survey item no.	Strongly disagree/ disagree		Somewhat disagree		Somewhat agree		Strongly agree/agree	
	Pre %	Post %	Pre %	Post %	Pre %	Post %	Pre %	Post %
*1. I can learn to code	1.6	1.6	4.1	3.5	14.9	12.4	79.4	82.5
**2. I am good at coding	19.0	9.5	11.7	8.9	40.6	34.6	28.6	47.0
3. I am good at problem solving	2.5	2.8	3.8	6.7	31.4	32.4	62.2	58.1
*4. I can write clear instructions for robot or computer to follow	8.6	8.9	16.5	7.0	32.7	30.5	42.2	53.6
5. If my code doesn't work, I can find my mistake and fix it	3.5	2.5	7.3	5.4	27.6	27.6	61.6	64.4
*6. I've been told I would be good at coding	21.3	20.6	19.4	13.3	25.1	27.0	34.3	39.0
**7. I like coding, or I think I would like coding	4.1	9.2	2.2	8.3	20.0	17.1	73.7	65.3
**8. I would like to learn more about coding	4.1	9.8	2.9	8.9	11.1	20.0	81.9	61.3
**9. Solving coding problems seems fun	4.7	14.0	8.3	10.5	22.2	25.7	63.7	49.9
**10. Coding is interesting	4.8	7.3	3.2	5.4	14.3	19.4	77.8	67.9
**11. I would like to study coding in the future	15.6	25.1	15.2	13.7	28.3	24.1	41.0	37.1
12. I can use coding skills in other school subjects	10.1	9.5	11.1	13.0	27.3	30.2	51.4	47.3
**13. Knowing how to code will help me to create useful things	3.1	5.4	5.7	7.9	20.6	27.9	70.5	58.7
*14. Knowing how to code will help me to solve problems	3.5	6.7	8.3	10.8	26.0	26.3	62.2	56.2
**15. I think I will need to understand coding for my future job	17.4	24.1	16.5	18.4	27.9	24.8	38.1	32.7
16. My friends think coding is cool	7.0	9.2	14.6	10.2	36.8	33.3	41.6	47.3
17. My parents think coding is important	10.2	12.4	13.3	14.3	34.3	28.3	42.2	45.1
**18. I am friends with kids who code	15.8	7.6	12.4	10.5	24.4	21.3	47.3	60.6
19. Kids who code are smarter than average	22.9	20.4	17.1	19.0	30.8	31.7	29.2	28.9
20. Kids who code enjoy doing sports	31.2	27.3	27.3	29.2	27.6	27.6	14.0	15.8
21. Coders are good at math	5.4	6.3	8.6	10.2	31.7	30.8	54.3	52.7
*22. Coders are good at science	5.4	5.4	9.8	15.6	33.7	39.4	51.1	39.7
23. Coders are good at language arts	14.3	14.9	23.8	25.4	34.0	37.1	27.9	22.6

Note: $^{*}p < 0.05$; $^{**}p < 0.001$.

Programmers can be most effective when they have *both* the needed technical skills and the communication and planning skills to effectively convey their intended meaning. Second, previous research (e.g., Taimalu and Luik 2019; Spiteri and Chang Rundgren 2020) shows that teachers are most likely to integrate new technologies and technological concepts when they find them acceptable and useable, and would be beneficial to their students. The high usability ratings from teachers for the CoCo platform and lessons is promising for future implementation.

Finally, the results also show positive changes from students. First, as hoped, students' computational thinking skills improved from pre-to post-test. Although computational thinking was integrated throughout all lessons and is required to work in CoCo itself, each instructional unit highlighted a specific computational thinking skill and included activities to help students practice that skill. Thus, the CoCo platform and lessons may be useful for improving computational thinking skills. Further, results show that after participating in the lessons, a greater number of students agreed that they could learn to code, were good at coding, had other people tell them they would be good at coding, could write good instructions for a robot or computer, and were friends with kids who code. These results indicate that participation may have improved students' self-efficacy for coding. However, there were also some surprising findings. First, there was about an 8% decrease in the number of students indicating that they agree or strongly agree that they like coding or think they would like coding. Although there is no way to know why students responded this way, we hypothesize that it may be because students encountered problems while coding and learned that coding can be difficult, whereas they may previously have thought of it as fun activity without considering the problem-solving required when coding. Relatedly, there was also a decrease in the number of students strongly agreeing that solving coding problems seems fun; yet, there was an increase in the number of students somewhat agreeing that solving coding problems can be fun. Together, these findings may support our hypothesis that students encountered difficulties when coding, which may have given them a more realistic picture of the challenging nature of coding. Thus, future research should focus on how students address challenges and how it influences their perception of coding. There were also small decreases in the percentage of students strongly agreeing that they would like to study coding in the future and that knowing how to code will help them solve problems. One possible explanation for these findings is that, although our lessons had a career component that featured videos from real-life programmers, teachers indicated that they often omitted these videos from the lessons because they ran out of time. As designed, the videos were supposed to be played at the end of class after students had completed their projects for the day, but teachers reported that students rarely finished at the same time. Thus, the research team plans to redesign the lessons to include the career connections at the beginning to ensure that it is included. Additionally, future research should emphasise career connections and interesting ways that coding is used in a variety of jobs.

The current study has several limitations. First, the relatively small number of participating teachers ($n = 11$) limits the generalizability of the findings regarding teacher perceptions and experiences with the CoCo platform and curriculum. While the student sample size ($n = 595$) is robust, the participants were drawn from a single, albeit diverse, school. This limitation restricts the extent to which the results can be generalised to other educational contexts. Furthermore, the study relied on self-reported data from both teachers (reflections and interviews) and students (attitudes survey), introducing the potential for bias and subjective interpretation. The use of existing assessment instruments, while established in the field, may not perfectly capture the specific nuances of computational thinking development fostered by the CoCo platform. Finally, the study's duration may not fully capture the long-term effects of CoCo on students' coding skills and attitudes; longitudinal research would be necessary to assess sustained impact. Further investigation is warranted to explore the generalizability of these findings across diverse geographical locations, student demographics, and with larger teacher samples.

6 | Conclusion

Results from this study lead us to conclude that a scaffolded, structured, and integrated approach for teaching computer science and writing can be highly useful and beneficial for students in the elementary grades. Not only did students' computational thinking skills and self-efficacy for coding improve in some areas, but teachers were highly satisfied with the CoCo platform and curriculum. The CoCo platform and curriculum were designed to be inclusive of a wide range of students to enable broader participation, and teachers indicated that indeed it was. This is an important step forward since previous research indicates that students with disabilities have fewer opportunities to use digital devices, and fewer opportunities to learn CS and digital literacy skills (Clendon and Erickson 2008; Hutchison et al. 2021). This lack of opportunity may explain why teachers reported that some of their students with learning disabilities had difficulty logging into their CoCo accounts. Thus, we view the CoCo platform and lessons as a positive approach for providing exposure to CS, and digital technologies in general, to students who may not otherwise have access or exposure. Most importantly, students widely made use of the features of the CoCo platform and used Scratch to code digital products that clearly communicated their message. This result indicates that, with appropriate scaffolding, integrating coding and writing can be an empowering approach for students to share their ideas and have their voice heard.

Author Contributions

Amy Hutchison: conceptualization, investigation, funding acquisition, writing – original draft, methodology, project administration, supervision. **Qi Si:** methodology, data curation, writing – review and editing, formal analysis. **Jamie Colwell:** investigation, writing – review and editing, methodology, conceptualization. **Erdogan Kaya:** conceptualization, investigation, methodology, writing – review and editing, software. **Eileen Jakeway:** writing – original draft, project administration, investigation. **Brittany Miller:** investigation, resources. **Kristie Gutierrez:** conceptualization, investigation, writing – original draft. **Kelly Regan:** conceptualization, investigation. **Anna Evmenova:** conceptualization, investigation.

Ethics Statement

This research has been approved by the George Mason University's Institutional Review Board (no. 1716624-4).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Asbell-Clarke, J., E. Rowe, V. Almeda, et al. 2021. "The Development of Students' Computational Thinking Practices in Elementary-and Middle-School Classes Using the Learning Game, Zoombinis." *Computers in Human Behavior* 115: 106587. https://doi.org/10.1016/j.chb.2020.106587.

Ayman, R., N. Sharaf, G. Ahmed, and S. Abdennadher. 2018. "MiniColon; Teaching Kids Computational Thinking Using an Interactive Serious Game." In *Joint International Conference on Serious Games*, edited by S. Göbel et al., 79–90. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-02762-9_9.

Bers, M. U. 2020. *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom*. New York: Routledge.

Bers, M. U., L. Flannery, E. R. Kazakoff, and A. Sullivan. 2014. "Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum." *Computers & Education* 72: 145–157. https://doi.org/10.1016/j.compedu.2013.10.020.

Casey, J. E., P. Gill, L. Pennington, and S. V. Mireles. 2018. "Lines, Roamers, and Squares: Oh My! Using Floor Robots to Enhance Hispanic Students' Understanding of Programming." *Education and Information Technologies* 23: 1531–1546. https://doi.org/10.1007/s10639-017-9677-z.

Clendon, S. A., and K. A. Erickson. 2008. "The Vocabulary of Beginning Writers: Implications for Children With Complex Communication Needs." *Augmentative and Alternative Communication* 24, no. 4: 281–293. https://doi.org/10.1080/07434610802463999.

Colwell, J., A. Hutchison, K. Gutierrez, J. Offutt, and A. Evmenova. 2023. "Elementary Teachers' Experiences in Online Professional Development for Literacy-Focused Computer Science Instruction for All Learners." *Computer Science Education* 34, no. 3: 546–565. https://doi.org/10.1080/08993408.2023.2263831.

Deniz, H., E. Yesilyurt, E. Kaya, A. Newley, and E. Lin. 2020. "Integrating Engineering, Science, Reading, and Robotics Across Grades 3–8 in a STEM Education Era." In *Proceedings of Society for Information Technology & Teacher Education International Conference*, edited by D. Schmidt-Crawford, 885–891. Waynesville, NC: Association for the Advancement of Computing in Education (AACE). https://www.learntechlib.org/primary/p/215840/.

DiCecco, V. M., and M. M. Gleason. 2002. "Using Graphic Organizers to Attain Relational Knowledge From Expository Text." *Journal of Learning Disabilities* 35, no. 4: 306–320. https://doi.org/10.1177/002221940203500402 01.

Egbert, J., S. A. Shahrokni, R. Abobaker, and N. Borysenko. 2021. "'It's a Chance to Make Mistakes': Processes and Outcomes of Coding in 2nd Grade Classrooms." *Computers & Education* 168: 104173. https://doi.org/10.1016/j.compedu.2021.104173.

Evmenova, A. S., K. Regan, A. Boykin, et al. 2016. "Emphasizing Planning for Essay Writing With a Computer-Based Graphic Organizer." *Exceptional Children* 82, no. 2: 170–191. https://doi.org/10.1177/0014402915591697.

Hutchison, A., J. Colwell, K. Gutierrez, A. Evmenova, J. Offutt, and V. Taylor. 2021. "Designing a Model of Computer Science Professional Development for Elementary Educators in Inclusive Settings." *Journal of Technology and Teacher Education* 29, no. 2: 165–193.

Hutchison, A., J. Colwell, Q. Si, K. Regan, K. Guttieriz, and E. Kaya. 2023. "Combatting the Flashy but Fuzzy Effect in Multimodal Writing With the Compose and Code Platform." Paper presented at the annual conference of the Literacy Research Association, Atlanta, GA.

Hutchison, A., A. Evmenova, K. Regan, and B. Gafurov. 2024. "Click, See, Do: Using Digital Scaffolding to Support Persuasive Writing Instruction for Emerging Bilingual Learners." *Reading Teacher* 77, no. 6: 810–811. https://doi.org/10.1002/trtr.2310.

Hutchison, A., L. Nadolny, and A. Estapa. 2016. "Using Coding Apps to Support Literacy Instruction and Develop Coding Literacy." *Reading Teacher* 69, no. 5: 493–503. https://doi.org/10.1002/trtr.

Katterfeldt, E.-S., M. Cukurova, D. Spikol, and D. Cuartielles. 2018. "Physical Computing With Plug-and-Play Toolkits: Key Recommendations for Collaborative Learning Implementations." *International Journal of Child-Computer Interaction* 17: 72–82. https://doi.org/10.1016/j.ijcci.2018.03.002.

Kistner, S., K. Rakoczy, B. Otto, C. Dignath-van Ewijk, G. Büttner, and E. Klieme. 2010. "Promotion of Self-Regulated Learning in Classrooms: Investigating Frequency, Quality, and Consequences for Student Performance." *Metacognition and Learning* 5: 157–171. https://doi.org/10.1007/s11409-010-9055-3.

Lee, C. C., and S. C. Tan. 2010. "Scaffolding Writing Using Feedback in Students' Graphic Organizers–Novice Writers' Relevance of Ideas and Cognitive Loads." *Educational Media International* 47, no. 2: 135–152.

Li, Y., A. H. Schoenfeld, A. A. diSessa, et al. 2020. "Computational Thinking Is More About Thinking Than Computing." *Journal for STEM Education Research* 3: 1–18.

Mason, S. L., and P. J. Rich. 2020. "Development and Analysis of the Elementary Student Coding Attitudes Survey." *Computers & Education* 153, no. 103: 898. https://doi.org/10.1016/j.compedu.2020.103898.

Master, A., S. Cheryan, A. Moscatelli, and A. N. Meltzoff. 2017. "Programming Experience Promotes Higher STEM Motivation Among First-Grade Girls." *Journal of Experimental Child Psychology* 160: 92–106.

Mayer, R. E., and R. Moreno. 2003. "Nine Ways to Reduce Cognitive Load in Multimedia Learning." *Educational Psychologist* 38, no. 1: 43–52. https://doi.org/10.1207/S15326985EP3801_6.

Murcia, K., C. Pepper, M. Joubert, E. Cross, and S. Wilson. 2020. "A Framework for Identifying and Developing Children's Creative Thinking While Coding With Digital Technologies." *Issues in Educational Research* 30, no. 4: 1395–1417.

Novak, E., and J. I. Khan. 2022. "A Research-Practice Partnership Approach for Co-Designing a Culturally Responsive Computer Science Curriculum for Upper Elementary Students." *TechTrends* 66, no. 3: 527–538. https://doi.org/10.1007/s11528-022-00730-z.

Parker, M. C., Y. S. Kao, D. Saito-Stehberger, et al. 2021. "Development and Preliminary Validation of the Assessment of Computing for Elementary Students (ACES)." In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 10–16. https://doi.org/10.1145/3408877.3432376.

Piliouras, T., R. Yu, K. Villanueva, et al. 2014. "A Deeper Understanding of Technology Is Needed for Workforce Readiness—Playing Games, Texting, and Tweets Aren't Enough to Make Students Tech-Savvy." In *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, 1–8. IEEE. https://doi.org/10.1109/ASEEZone1.2014.6820656.

Ponce, H. R., R. E. Mayer, and M. J. Lopez. 2013. "A Computer-Based Spatial Learning Strategy Approach That Improves Reading Comprehension and Writing." *Educational Technology Research and Development* 61: 819–840. https://doi.org/10.1007/s11423-013-9310-9.

Sanders, S., M. Losinski, R. Parks Ennis, W. White, J. Teagarden, and J. Lane. 2019. "A Meta-Analysis of Self-Regulated Strategy Development Reading Interventions to Improve the Reading Comprehension of Students With Disabilities." *Reading & Writing Quarterly* 35, no. 4: 339–353. https://doi.org/10.1080/10573569.2018.1545616.

Shute, V. J., C. Sun, and J. Asbell-Clarke. 2017. "Demystifying Computational Thinking." *Educational Research Review* 22: 142–158. https://doi.org/10.1016/j.edurev.2017.09.003.

Spiteri, M., and S. N. Chang Rundgren. 2020. "Literature Review on the Factors Affecting Primary Teachers' Use of Digital Technology."

Technology, Knowledge and Learning 25: 115–128. https://doi.org/10.1007/s10758-018-9376-x.

Sturm, J. M., and J. L. Rankin-Erickson. 2002. "Effects of Hand-Drawn and Computer-Generated Concept Mapping on the Expository Writing of Middle School Students With Learning Disabilities." *Learning Disabilities Research & Practice* 17, no. 2: 124–139. https://doi.org/10.1111/1540-5826.00039.

Su, S. W., L. X. Chen, S. M. Yuan, and C. T. Sun. 2024. "Cultivating Creativity and Improving Coding Skills in Primary School Students via Domain-General and Domain-Specific Learning Scaffoldings." *Education Sciences* 14, no. 7: 695.

Sullivan, A., and M. U. Bers. 2016. "Robotics in the Early Childhood Classroom: Learning Outcomes From an 8-Week Robotics Curriculum in Pre-Kindergarten Through Second Grade." *International Journal of Technology and Design Education* 26: 3–20. https://doi.org/10.1007/s10798-015-9304-5.

Taimalu, M., and P. Luik. 2019. "The Impact of Beliefs and Knowledge on the Integration of Technology Among Teacher Educators: A Path Analysis." *Teaching and Teacher Education* 79: 101–110. https://doi.org/10.1016/j.tate.2018.12.012.

Thomas, D. R. 2006. "A General Inductive Approach for Analyzing Qualitative Evaluation Data." *American Journal of Evaluation* 27, no. 2: 237–246. https://doi.org/10.1177/1098214005283748.

Vee, A. 2017. *Coding Literacy: How Computer Programming Is Changing Writing*. Cambridge, MA: MIT Press.

Vegas, E., M. Hansen, and B. Fowler. 2021. "Building Skills for Life: How to Expand and Improve Computer Science Education Around the World." Technical Report. Center for Universal Education at Brookings. https://www.brookings.edu/articles/building-skills-for-life-how-to-expand-and-improve-computer-science-education-around-the-world/.

Yesilyurt, E., E. Kaya, R. Turgut, E. Adibelli-Sahin, B. Sahin, and H. Deniz. 2022. "Teaching Elementary Computer Science With Physical Computing for Linguistically Diverse Classrooms." In *Proceedings of Society for Information Technology & Teacher Education International Conference*, edited by E. Langran, 1950–1955. San Diego, CA, United States: Association for the Advancement of Computing in Education (AACE). https://www.learntechlib.org/primary/p/221007/.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.