IJCSES

Volume 6, No: 2
October 2023
ISSN 2513-8359

International Journal of Computer Science Education in Schools

Editors

Dr Filiz Kalelioglu

Dr Yasemin Allsop

www.ijcses.org

International Journal of Computer Science Education in Schools

October 2023, Vol 6, No 2

DOI: 10.21585/ijcses.v6i2

Table of Contents

‘Guyanese Girls Code’ Goes Virtual: Exploring Instructors’ Experiences

Penelope Defreitas[1]
Alicia Layne[1]

[1]University of Guyana, Department of Computer Science

DOI: https://doi.org/10.21585/ijcses.v6i2.168

Abstract

The Guyanese Girls Code (GGC) training program, established in 2018, is aimed at increasing female participation in ICT. As a result of the COVID-19 pandemic, the program shifted to virtual operations to ensure the safety of students and instructors. This presented an opportunity to contribute to the growing body of research that has been investigating the virtual implementation of such ICT interventions. Additionally, potential value was recognized in examining the instructors’ adoption of the GGC’s teaching model to the virtual mode. The program delivered the GGC curriculum primarily via the Scratch programming environment. It involved 80 female students between the ages of nine (9) and fourteen (14), and six (6) instructors. Data was collected via a focus group discussion involving three (3) instructors who shared their experiences of the virtual program. Also, data from the program's student survey was used to gain an understanding of the students’ background and to enhance the narrative about the recent iteration of the GGC program. It was found that mentorship and fostering a community of learners were positive extensions of the instructors’ role. Further, game-based activities, live demonstrations, breakout rooms and projects were observed to be effective strategies in delivering the program virtually. However, parent-driven enrollment, some aspects of the virtual learning environment and the use of flowcharts for problem solving proved to be challenging. Recommendations were made for future iterations of the GGC program and other similar interventions.

Keywords: Scratch programming, instructor experiences, school children, teaching model, virtual learning environment

1. Introduction

1.1 Support for Guyanese Girls in ICT

The Guyanese Girls Code program is an ICT training program aimed at introducing females from grades seven (7) to nine (9) to the field of ICT, with special emphasis on skills like problem solving and critical thinking. The program stemmed from low enrollment and graduation statistics of females in the Computer Science Department at the University of Guyana (UG), as well as the global underrepresentation of females in the ICT field.

From 2018 to 2020, the then Ministry of Public Telecommunications (MoPT) and the National Center for Educational Resource Development (NCERD) collaborated with UG to implement the GGC program (DPI, 2019). The mission of the program was furthered in 2021 through Guyana's Office of the Prime Minister, Industry, and Innovation Unit (IIU). Through the GGC and similar programs, Guyana remains committed to realizing the United Nations (UN) 2030 Agenda for Sustainable Development in the areas of gender equality, reduced inequalities, and quality education.

Since 2018, numerous females have benefited from strategic ICT training initiatives in Guyana. To date, approximately 175 female students have completed the GGC program. Many others continue to gain ICT-related skills via national code camps, which aim to further reduce inequalities by targeting students from remote regions of the country (DPI, 2021a; DPI, 2021c). Even during the COVID-19 pandemic, these training projects have persisted. Some were run remotely, while others were conducted face-to-face - adhering to COVID-19 safety measures.

Given that face-to-face programs were conducted at facilities equipped with internet-ready devices, from a resource perspective, they were highly accessible to students. However, due to budgetary constraints and the sudden adoption of virtual delivery at the onset of the pandemic, students opting for the virtual mode were required to have internet-enabled laptops or desktop computers to participate in the program. While the option to enroll in the more accessible face-to-face mode remained, an opportunity was presented to pilot virtual delivery of the GGC program.

1.2 Underrepresentation of Women in ICT

The underrepresentation of women in STEM fields continues to be a global concern for governments and international organizations. Through initiatives such as the 'STEM and Gender Advancement' (SAGA) project and the 'EQUALS Global Partnership for Gender Equality in the Digital Age', governments and policymakers worldwide have been supported in boosting women's visibility, participation and recognition in STEM. Females have been reported to represent only 35% of global enrollment in STEM-related studies at the tertiary level - with notably lower enrollment in disciplines related to ICT (UNESCO, 2017). In 2021, for instance, the European Union reported that only 17% of ICT specialists in its member countries were female (European Commission, 2021); while data from UG revealed that from 2009-2019, the number of female graduates with computing degrees was consistently lower than male graduates (Layne et al., 2020).

These trends are cause for concern because it has been estimated approximately 90% of contemporary jobs are likely to require ICT skills - leaving the possibility of some 1 million unfilled ICT vacancies (UNESCO, 2017). Female underrepresentation in this area is therefore not only damaging to equity, but also presents the risk for current and upcoming shortages and imbalances in the labor market (OECD, 2018). In response, international organizations such as the ITU, UNESCO and EQUALS Global Partnership support and advocate for a concerted effort among governments, the private sector,

researchers and other communities in providing a gender-responsive approach to ICT education and representation (ITU, 2021; UNICEF, 2020; OECD, 2018; UNESCO, 2017).

1.3 COVID-19 and Shifting to Virtual Training

The COVID-19 pandemic resulted in a widespread shift to a range of virtual educational strategies (e.g., paper-based take-home packages, television or radio programs, phone calls, tutoring and online platforms) to keep students and teachers safe (Li & Lalani, 2020; UNESCO, 2020). As a result, teachers across the globe swiftly adapted their learning materials and teaching strategies into formats that were suitable for virtual engagement to ensure that their educational efforts and impact persisted. The GGC program also adopted virtual training in an effort to continue safe operations during the COVID-19 pandemic. However, the rapid transition proved challenging because the program's teaching model was tailored to the face-to-face mode of delivery (Layne et al., 2020).

Further, the literature on teaching programming online as of March 2020 was primarily related to higher education settings (McDonald & Dillon, 2021) and massive open online courses, and therefore not readily applicable to the engagement of younger learners in virtual environments (e.g., Skalka et al., 2019; Robinson & Carroll, 2017; Staubitz et al., 2016). While more recent work has emerged to fill this gap (e.g., Benvenuti et al., 2021; Garcia-Ruiz et al., 2021; McDonald & Dillon, 2021), at the height of the COVID-19 pandemic the GGC program encountered a vacuum of knowledge on appropriate techniques and strategies for virtual delivery of programming curricula to younger learners. Nonetheless, the GGC program was successfully completed with 70 out of the 80 students graduating and providing positive feedback.

Considering this successful adoption of a new mode of delivery, we recognize the potential to gain valuable insights from the instructors' experiences. Of interest in particular is how the GGC teaching model - developed to inform the program's curriculum design and teaching strategies in the face-to-face mode (Layne et al., 2020) - was implemented by the instructors for online operations. Insights gained from this study can provide guidance for similar ICT training inventions and inform the extension of the GGC teaching model.

2. Overview

The GGC program's teaching model was originally designed for face-to-face delivery. During the COVID-19 pandemic, it was used in the virtual mode for the first time. In this study, we aim to determine instructors' experiences in adopting the model to this new mode of operations. Our research questions are:

a) How were the components of the teaching model adopted by the GGC instructors?

b) What challenges did the GGC instructors face in adopting the teaching model?

We begin by reviewing the literature on virtual learning engagement and the experience of similar ICT training interventions during the COVID-19 pandemic. The implementation details of the GGC program prior to and during the pandemic are then provided, as well as this study's data collection and analysis techniques. Finally, major findings are discussed.

3. Literature Review

3.1 Fostering Virtual Learning Engagement

Virtual learning engagement typically occurs synchronously or asynchronously. Synchronous methods are delivered live, using communication software with features such as audio, video, text chat, interactive whiteboard, and screen sharing (Lim, 2017; Martin & Parker, 2014). In addition, breakout rooms may be employed for facilitating small group discussions. On the other hand, asynchronous methods are usually facilitated via a learning management platform (UNESCO, 2020; Lim, 2017), which provides mechanisms for students to access learning materials, receive notifications, complete activities and communicate with peers (Lim, 2017). Some training programs are run using either synchronously or asynchronously, whereas others utilize a combination of the two approaches. Regardless of the strategy, four core requirements are needed to facilitate robust virtual learning engagement: high-speed internet service, internet-enabled devices, instructional content, and support such as digital literacy, teacher readiness and technical assistance (Chandra et al., 2020).

There is value in blending the synchronous and asynchronous virtual learning approaches (Yamagata-Lynch, 2014), especially when transitioning from the face-to-face mode (Fadde & Vu, 2014). The synchronous technique mirrors face-to-face classrooms to an extent since it allows live interaction with teachers and peers, but this method becomes difficult to manage when the class size is large (Lim, 2017). Synchronous learning may be coupled with the asynchronous method since learning can be further supported outside of live sessions through access to learning materials, notifications and a network of teachers and peers. However, some major disadvantages of asynchronous learning are delayed feedback, irregular student participation in activities, and notifications or written instructions that are subject to interpretation (Lim, 2017). Nonetheless, students can benefit from the strengths of blending the synchronous and asynchronous learning approaches (Yamagata-Lynch, 2014). In particular, they may be able to better stay on task, gain a sense of stability and develop a stronger connection with peers when engaged in discussions.

Researchers have studied and recommended strategies to improve the virtual learning engagement experience. For instance, Chen et al. (2020) revealed that students had a strong preference for live and pre-recorded lectures alongside synchronous complementary discussions. In addition, engagement activities such as question and answer, small group case study discussions (Chen et al., 2020; Martin & Parker, 2014) and quizzes (Chen et al., 2020; Skylar, 2009) during live sessions were found to encourage engagement. A similar study, involving a larger cross-section of students found that for learner-to-learner engagement, activities such as icebreakers, collaborative work, peer presentations,

and peer review of assignments were perceived as valuable (Bolliger & Martin, 2018). Additionally, for learner-to-instructor engagement, regular communication through emails, announcements, reminders, and discussions were deemed important. Furthermore, for learner-to-content engagement, the provision of structured discussions, realistic scenarios, and content in multiple media formats were highly valued. Overall, the least valued activities included synchronous guest talks, events and self-tests.

3.2 ICT Training Interventions

Numerous ICT training interventions for young people have been motivated by low participation within marginalized communities and the slow integration of computing education into the formal school curriculum (Alsheaibi et al., 2020). Supported by universities and/or the public sector, these initiatives are typically conducted as after-school programs and address various social barriers to ICT (e.g., Spartan Girls Who Code (McDonald & Dillon, 2021), Guyanese Girls Code (Layne et al., 2020) and GreekCodersK12 (Misthou et al., 2021)). As a result, they play an important role in making the field more accessible to groups that may be disproportionately affected by limited formal ICT training opportunities (Wang & Moghadam, 2017; Goode, 2008).

Aiming to serve as an entry point to computing and coding, the curricula of these interventions tend to be centered on the fundamentals of computing and computer programming. They commonly use the Scratch programming language, as well as physical computing kits such as the BBC micro:bit, Arduino and Lego Mindstorms (Alsheaibi et al., 2020). In some programs, the curriculum is also extended to develop students' critical thinking skills and awareness of various social and environmental issues (e.g., Kafai et al., 2021; Misthou et al., 2021). Studies have also focused on the teaching practices and engagement strategies adopted, and the experience and perceptions of the students and/or program instructors in the face-to-face mode (e.g., Alsheaibi et al., 2020; Layne et al., 2020; Aivaloglou & Hermans, 2019; Burke & Kafai, 2010).

However, during the COVID-19 pandemic, researchers began to prioritize the investigation of these initiatives in the virtual mode. For instance, McDonald and Dillon (2021) captured the experience of the Spartan Girls Who Code club as it transitioned to virtual engagement during the pandemic. The club, supported by the students and faculty of Michigan State University, aimed to introduce computing to young female students. In the abrupt transition to virtual engagement, they found technologies and platforms such as Zoom, CodeHS, Google Docs and Remind particularly useful in connecting with students and their parents. Live coding, virtual coding exercises and projects were also key in conducting lessons and assessments. Additionally, in their experience, virtual icebreakers, games, show-and-tell and opportunities for student reflection were also crucial to engagement.

Similarly, Krug et al. (2021) analyzed the results of the 'CodeBeats' camp that was conducted virtually during the pandemic. The intervention leveraged hip-hop, musical coding software and scaffolded exercises to introduce computer programming to minority middle grade students. While

this study was not explicitly aimed at distilling lessons learnt from virtual engagement, it is noted that technologies such as Twitch and Mentimeter along with frequent quizzes and creative online classes were used to deliver content and engage the students. These classes adopted the style of a 'news show' through live segments that introduced more detailed pre-recorded sessions.

4. The GGC Program

4.1 The Teaching Model

The GGC program targets females from grades seven (7) to nine (9). It is geared at introducing them to the field of ICT, with special emphasis on 21st century competencies such as problem solving and critical thinking. The 'Motivation, Support and Teaching Components' tree model (MST-tree model) (see Figure 1), developed in the first iteration of the GGC program, informs the curriculum design and teaching strategies used to deliver the ICT training (Layne et al., 2020).

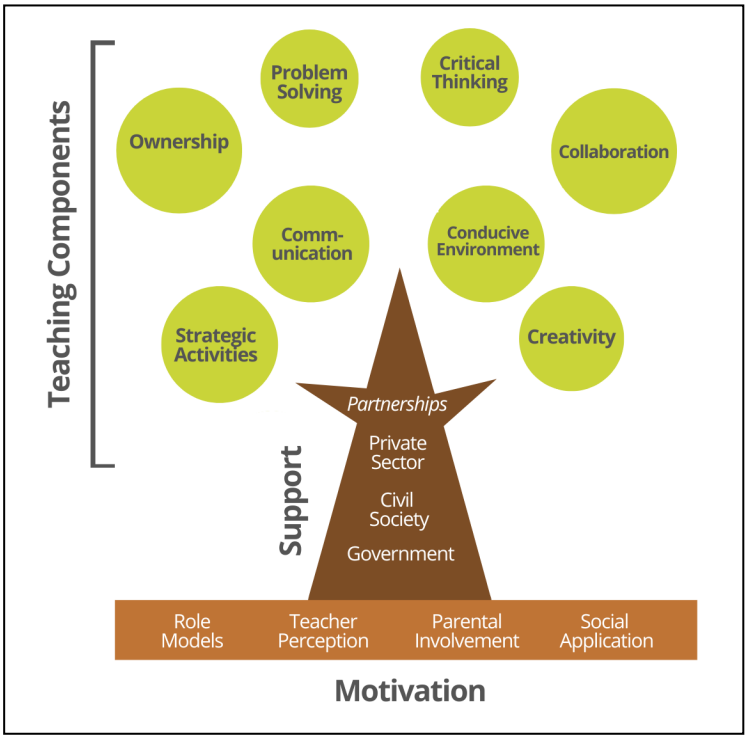


Figure 1. MST-tree model

Taking the form of a tree, the model presents a metaphor for development in the field of ICT. It prioritizes Motivation, Support and Teaching, and outlines elements and strategies within each component. Layne et al. (2020) reported on the positive impact of these components in a previous GGC iteration, thereby lending support to the model's adoption in future programs.

For example, the high levels of self-efficacy and interest generated in the first GGC iteration were attributed to elements within the model's Motivation component. The literature has also more

generally identified these elements – inclusive of role models (Stelter et al., 2021; Stoeger et al., 2013), positive teacher perception (Vekiri, 2010), parental involvement (Jungert et al., 2020; Šimunović & Babarović; 2020; Šimunović et al., 2018) and opportunities to explore the social applications of ICT (Vekiri, 2010) – as valuable to children's interest and motivation in ICT.

The Support component of the model was also recognized as a critical enabler for the program. Its role in the overall facilitation of the ICT intervention through the fostering of a conducive environment and ensuring access to training opportunities were notable observations in Layne et al.'s (2020) study of the initial GGC iteration. Furthermore, the Support component's emphasis on partnerships among government, civil society, and the private sector aligns with international advocacy for concerted efforts toward inclusive education and representation.

The Teaching components of the model were proposed because of their potential to effectively deliver ICT curricula and to adequately prepare students for the 21st century. In particular, components such as creativity, problem solving and critical thinking were identified as fundamental skills for promoting active participation in the world of work (Voogt & Roblin, 2012, Trilling & Fadel, 2009). Further, to expose these skills to young people, strategic activities involving tools like Scratch (Oluk & Korkmaz, 2016; Oh et al., 2013) and the BBC micro: bit (Abonyi-Tóth & Pluhár, 2019; Micro:bit Research) were found to be highly effective.

4.2 Transition to Virtual Operations

From the researchers' preliminary investigations of the program, the following subsections detail how the program was designed for virtual delivery:

4.2.1 Implementation and Curriculum

Prior to the COVID-19 pandemic, the GGC program was run as twelve (12) weekly face-to-face sessions. The sessions lasted for four (4) hours each and were conducted between the April to July school period. The program's curriculum comprised three (3) modules (see Table 1), with the first module - Computer Fundamentals and Scratch Programming, focusing on topics such as the fundamentals of hardware and software, female pioneers in computing, ethics in computing, fundamentals of algorithms, problem solving (e.g., narratives, pseudocode), programming fundamentals, the Scratch interface, and code blocks in the major categories (e.g., Events, Looks, Motion, Control, Variables, Operators, Sensing). The BBC micro:bit module, on the other hand, explored the physical features of the micro:bit (e.g., buttons, accelerometer, radio and Bluetooth antenna, processor, temperature sensor), whereas the HTML and CSS module introduced the area of web development. Nine (9) out of the twelve (12) sessions covered the program's curriculum via unplugged (e.g., My Robotic Friends) and plugged activities (e.g., Hour of Code). In addition, two (2)

sessions were devoted to creating a Scratch project, and one (1) session was set aside for a written and practical examination.

Table 1. GGC Curriculum modules

Curriculum modules	Duration	Pre-pandemic	During Pandemic
Computer Fundamentals and Scratch Programming	6 weeks	Yes	Yes
BBC micro:bit	2 weeks	Yes	No
HTML and CSS	1 week	Yes	No

During the pandemic, a few changes were made to the GGC's program implementation. For instance, the program was run virtually for eight (8) weekly sessions, between the July to August school break. Each session lasted for three (3) hours, and the students were allowed more breaks (e.g., ten (10) to fifteen (15) minutes after each hour) to reduce virtual meeting fatigue.

In terms of the curriculum, the virtual GGC program focused on the Computer Fundamentals and Scratch Programming module. The BBC micro:bit and the HTML and CSS modules were not included in this iteration of the program, due to the overall reduced delivery time. Moreover, the procurement process to obtain the BBC micro:bits proved to be challenging during the pandemic. Mirroring the face-to-face mode, the Computer Fundamentals and Scratch Programming module was covered in six (6) sessions using plugged and unplugged activities. In addition, the final two (2) sessions were devoted to creating a Scratch project. No written or practical examination was conducted; however, the students received credit for their attendance, homework activities, and the Scratch project. While the project allowed students flexibility to derive their own ideas, specific assessment guidelines were outlined. The students had a choice among the use of algorithms, pseudocode or flowcharts to support problem solving in the final project.

Flowcharting was a new addition to the program. It was taught using digital (i.e., slideshows, Zoom whiteboard) and paper-based methods (i.e., pen/pencil and paper). The slideshow was utilized for presenting an overview of the problem-solving approach and was complemented by the Zoom whiteboard for facilitating practical demonstrations and collaborative input from the students. The paper-based approach was employed for individual flowcharting activities, and therefore required photos of the diagrams to be uploaded to Google classroom. The instructors reused algorithms from previous sessions as a problem base for the flowchart demonstrations and activities. These were converted into flowcharts and presented side by side to draw comparisons.

4.2.2 Academic and Government Support

The GGC program curriculum was designed by personnel of the Computer Science Department, UG. During the virtual program, one (1) academic offered weekly guidance to the instructors by reviewing lesson plans and instructional materials and offering advice on challenges that emerged. Government support took the form of coordinating the student recruitment process, offering technical assistance and Zoom access, providing stipends to instructors, rewarding the students, and ensuring the smooth running of the program.

4.2.3 Program Recruitment

In an effort to spread awareness about the GGC program and the recruitment process, advertisements were published via social media, local newspapers, and government websites (e.g., DPI, 2021b). These advertisements targeted parents and guardians who were responsible for submitting applications on behalf of their children.

Due to budgetary constraints and the sudden adoption of virtual engagement at the onset of the pandemic, students opting for the virtual GGC program were required to have internet-enabled laptops or desktop computers. Nonetheless, working with this group presented the opportunity to pilot virtual delivery, which would serve to inform future iterations of the program.

4.2.4 GGC Recruits and Instructors

Eighty (80) female students between the ages of nine (9) and fourteen (14), were shortlisted for the virtual GGC program. All students were digitally literate and had access to an internet-enabled laptop or desktop computer. The students were assigned to two groups, comprising 40 members each. A preliminary survey was conducted with parental consent to gather information on the students' prior knowledge, expectations, perceptions about ICT, etc.

A total of six (6) female instructors, three (3) per student group, were involved in the GGC program. Each instructor possessed a Bachelor's degree in Computer Science. In addition, several of them served as collaborators on community projects and laboratory demonstrators for introductory programming courses at the Computer Science Department, UG.

With assistance from the Computer Science Department, UG, the instructors collaboratively prepared lesson plans, activities, and instructional materials for the program. This strengthened the program's delivery and ensured consistent facilitation across the student groups. During the sessions, the instructors delivered the curriculum, demonstrated practical examples, and offered guidance to the students. They also logged the strengths and weaknesses of each session which were discussed at the weekly planning meetings. To address the challenges faced, the team brainstormed possible solutions and created responsive action plans.

4.2.5 Learning Environment

The virtual program was supported by Zoom, Google Classroom, Gmail and WhatsApp. These platforms were used because a majority of the students and their parents or guardians were already familiar with them.

Zoom was used to facilitate the weekly synchronous engagement sessions, with features such as audio, video, share screen, reactions, chat, whiteboard, and breakout rooms being more commonly utilized. Meanwhile, asynchronous engagement occurred via Gmail and Google classroom (e.g., access to instructional materials, assessments, reminders, grade book). As added support, WhatsApp groups were utilized for announcements (e.g., homework and assessment reminders, meeting links, etc.), while direct messages and calls were exchanged between instructors and students (parents and/or guardians in some cases) for the purpose of check-ins and assistance with specific issues (e.g., technical challenges, follow-up questions on topics).

5. Method

Recognizing potential value in examining the implementation of online ICT training interventions, this study aimed to determine the experiences of GGC instructors in adopting the MST-tree model to the virtual mode of operations. This signalled the need for a qualitative study that would allow the researchers to holistically investigate the GGC program and focus on the instructors' subjective perspectives and experiences. Given the key components of the model, the investigation placed emphasis on the instructors' approach to motivating and engaging the students online, the challenges that emerged and the type of support received from the GGC stakeholders.

5.1 Participants

Of the six (6) GGC instructors, three (3) were purposively selected on the basis of their availability and central role in the planning and execution of the program. The instructors (P1, P2, P3) were engaged in a virtual focus group discussion that was centred on examining their experiences across key elements of the GGC teaching model.

Due to the small number of instructors, the researchers opted for one (1) focus group discussion. This approach also provided an opportunity for the instructors to jointly reflect on their individual and collective experiences in facilitating the program, especially since they were attached to different groups.

5.2 Procedure

The focus group discussion was moderated by the researchers who were involved in previous face-to-face iterations of the program. The discussion lasted approximately three (3) hours - with a fifteen (15) minute break in the middle of the discussion. During the discussion, the moderators alternated the

roles of note taker and lead moderator. The instructors were encouraged to speak freely and raise additional points. They were also informed that their participation was voluntary and that their responses would remain confidential.

The discussion was guided by semi-structured questions (see Appendix) related to the main components (i.e., Motivation, Support and Teaching) of the GGC teaching model (see Figure 1). These questions were aimed at exploring the instructors' virtual implementation of the components, as well as any challenges encountered. It is recognized that the moderators' past experiences with the face-to-face implementation of the model may have influenced their line of inquiry (see Merriam, 1988), but nonetheless presented an opportunity to further probe the instructors' responses (see Appendix).

Apart from the data that was collected via the instructor focus group, data from the preliminary GGC student survey (e.g., reasons for enrolment, prior knowledge, expectations, etc.), was utilized. The preliminary student survey data served as a supplementary resource to understand the students' background and to enhance the narrative about the recent iteration of the GGC program. Email consent was required from parents or guardians before the students participated in the preliminary survey. A listing of the survey questions was also provided to the parents to increase transparency.

5.3 Data Management and Analysis

The focus group's audio recording was converted to a verbatim textual transcript using an automated audio transcription service. The text was then manually cleaned, and each instructor was assigned a pseudonym.

The researchers then read the transcript multiple times to gain a high-level understanding of the data in its entirety. During this process, both researchers made preliminary notes about the data. In jointly reviewing their notes, it was recognized that substantial units of text in the transcript could be broadly categorized as 'Fact', 'Opinion', 'Recommendation', 'Challenge' and 'Additional Information'. Further, given the study's focus on the adoption of the GGC teaching model by the instructors, the model's primary components guided the creation of the predefined codes for data analysis (i.e., 'Motivation', 'Support' and 'Engagement Strategies'). It was recognized that the data surrounding the Teaching component of the GGC model suggested engagement of the students beyond the teaching context, and thus 'Engagement Strategies' was used as a more appropriate code.

To preserve the context of the coded text and to ensure that both expected and anomalous information could be captured (Creswell & Poth, 2016), the model-derived codes incorporated the preliminary categorizations noted by the researchers. The final coding framework included codes such as motivation-fact, motivation-opinion, motivation-additional-information, motivation-challenge, motivation-recommendation, etc.

The researchers then independently used the framework to undertake qualitative deductive coding. The Dovetail qualitative data analysis software assisted in the process of associating text segments in the transcript with codes from the framework. To allow for verification, the researchers identified and discussed differences between their coding to reach a consensus. The researchers sought to establish at least an 80% agreement (Miles & Huberman, 1994) on codes assigned to text segments. Following this, they mutually agreed on the core ideas and themes emerging from the coded data (see Table 2). The codes related to the Support component were excluded. This is because the supporting agents performed as expected and there was no need for further analysis and discussion.

The emerging themes formed the study's main findings and were analysed and discussed within the context of the GGC program and data collected from the preliminary survey. Following Creswell and Poth's (2016) recommended validation strategy, the researchers carefully factored the possible impact of their previous face-to-face GGC experience on the interpretation of this study's themes. To further reduce bias and strengthen the dependability of the findings, the researchers also relied on triangulation (Miles & Huberman, 1994; Lincoln & Guba, 1985) to find corroborating evidence and theories from the findings and recommendations of similar studies.

Table 2. Examples of the codes and emerging themes

Text Segment Examples	Codes	Emerging Themes
We not only tried to make the sessions relatable, and our examples practical and relatable, we also tried to make ourselves relatable.	motivation-fact	Mentorship and Learning Communities
And we would encourage them, like in the WhatsApp groups, if someone doesn't understand something, like allow the other girls to help them out instead of us just answering all the questions.	engagement-strategies-fact	
I would say that it was more the parents' idea than the girls. We struggled to sustain that interest.	motivation-challenge	Parental Involvement
I would have to reach out to and message and then I would get a response.	motivation-fact	

We need to get the parents more involved.	motivation-recommendation	
We used the Tower of Hanoi, which the girls solved like really quickly and they went above and beyond with that one.	engagement-strategies-fact	Effectiveness of Games
…at first we thought it [flowcharts] would have been simple, but it turned out to be extremely complex for the kids	engagement-strategies-challenge	Flowchart Challenges
...the implementation [final project] in Scratch did not reflect the problem. A few of them... were implementing projects that they didn't even [propose]	engagement-strategies-challenge	Final Scratch Project
...rather than telling them what to do, we literally demoed it... you have to do this like this, click that, you know... to help them overcome these hurdles.	engagement-strategies-fact	Virtual Engagement
...we say, just a second OK, we're coming to you... in a way to encourage them and keep that momentum going.	engagement-strategies-fact	Virtual Learning Environment Constraints

6. Discussion of Findings

This section discusses the findings on how the GGC teaching model (see Figure 1) was adopted by instructors in the virtual mode of operations. Of interest within the 'Motivation' component were mentorship and learning communities, as well as parental involvement. Furthermore, areas that stood out for the 'Teaching' component of the model included game-based activities, challenges surrounding flowcharts, effective virtual engagement strategies, Scratch-related assessments, and constraints of the virtual learning environment.

6.1 Mentorship and Learning Communities

Data collected from the GGC instructors suggest that motivation was fostered through the presentation of relatable role models to the students. It was found that instead of primarily relating stories of local

and international women in ICT for this purpose (Layne et al., 2020), the instructors more readily emphasized their personal experiences as professionals in the field.

> *We not only tried to make the sessions relatable, and our examples practical and relatable, we also tried to make ourselves relatable. (P1)*

The instructors' gravitation towards relatability signals a proactive attempt at extending their facilitation role to include role modelling and mentorship. There is considerable value in this since mentorships within STEM-focused activities have the potential to enhance students' science identity, self-efficacy, interest and commitment to pursuing related careers (Stelter et al. 2021; Stoeger et al., 2013). Further, in building the rapport required for the mentoring relationship by, for example, engaging in casual conversation (McReynolds et al., 2020) during breaks and after classes, the instructors may have been able to bridge their distance within the virtual environment and better position themselves to motivate the students.

> *Simple things like asking them about their day or what they're doing, like when we have breaks... they feel more comfortable and have better interaction. (P3)*

Additionally, the instructors' approach to 'mentoring' aligned with the effective many-to-many group mentoring structure (Stoeger et al., 2017) whereby students benefited from access to two or more instructors identified as mentors. Further, the instructors were observed to have encouraged learner-to-learner engagement not only inside, but also outside of the program's virtual classroom environment.

> *And we would encourage them, like in the WhatsApp groups, if someone doesn't understand something, like allow the other girls to help them out instead of us just answering all the questions. (P3)*

Collectively, these developments may have implications on extending the GGC teaching model to include the fostering of a learning community that provides the "structure for social interactions among students, their peers and STEM professionals" (Misthou, 2021, p. 956). Such an extension should also consider the training of prospective mentors to effectively create and sustain mentoring relationships (Stelter et al., 2021).

6.2 Parental Involvement

With respect to parental involvement as a source of motivation for the GGC students, the instructors observed that enrolment in the program may have been strongly influenced by the students' parents and guardians. This was corroborated by the program's initial survey data:

> *Honestly, my mom put me in this course and she didn't really give me an option, at first I thought it was a burden but then I told myself why not give it a shot. Also, this may be a benefit to my future so I accepted it. (GGC Student)*

While it has been found that parents' positive perceptions, enthusiasm and communication of STEM-related values are important in stimulating children's motivation in STEM (Šimunović et al., 2018;

Jungert et al., 2020), the instructors viewed their experience of this in the form of parent-driven enrolment as a possible cause for later challenges in sustaining the students' engagement and motivation.

> *I would say that it was more the parents' idea than the girls. We struggled to sustain that interest. (P2)*

This suggests that the dynamics of parental involvement in fostering children's motivation and interest should be carefully considered. As highlighted by Šimunović and Babarović (2020, p. 712), "parenting style, parents' support for a child's autonomy, and communication patterns... during coactivity" should be factored alongside parents' involvement in their children's educational and leisure activities in STEM-related fields. Therefore, as pointed out by the GGC instructors, parent-driven enrolment may have affected some students' autonomy in the program, which then reduced their sense of capability, interest and engagement. Further, primarily relying on parental involvement for fostering program awareness and enrolment may have excluded females whose parents are not aware of or interested in the field of ICT.

Collectively, these observations suggest the need to consider revision of the GGC program's recruitment strategies given that the current approach makes use of advertisements primarily targeted at parents (e.g., DPI, 2021b). It may be worth exploring additional recruitment strategies that are more inclusive and tailored to directly inviting females into STEM classrooms (e.g., outreach material featuring relatable female role models, Girls in STEM events or career fairs, conferences and collaboration with school counselors) (Shadding et al., 2016; Milgram, 2011).

6.3 Effectiveness of Games

The instructors used real-world scenarios, analogies and games to foster engagement. However, they observed that game-based activities generated the most interest among the students. This finding is similar to the study by Malliarakis et al. (2014), which reported that the use of games for teaching programming can provide a range of engaging characteristics (e.g., storytelling, scaffolding, interactivity), which positively impact student participation and encourage the completion of tasks through interesting scenarios.

During the earlier sessions, games were used to break the ice (e.g., The Fortune Teller [3]), encourage problem solving (e.g., TED-Ed Riddles[4], Tower of Hanoi[5]) and reinforce concepts in the GGC program.

[3] A turn-taking game, played in groups, for predicting the future of computing technology
[4] https://www.youtube.com/watch?v=7yDmGnA8Hw0
[5] https://www.mathsisfun.com/games/towerofhanoi.html

> *...rather than just explaining what problem solving is - the concept, we used the Tower of Hanoi, which the girls solved like really quickly and they went above and beyond with that one. (P1)*

Motivated by the enormous interest in game-based activities and students' preference for the creative elements of the Scratch environment (e.g., animating characters and creating personalized worlds) (Kalelioglu & Gülbahar, 2014), the instructors designed the later Scratch activities to allow students to create their own games.

> *I think the visual aspect of it [creating games using Scratch] was fun because it wasn't just programming with blocks. It was also the fact that they could see... their end goal while they're coding with blocks. It's really motivational. (P1)*

Impressively, the students made excellent Scratch submissions well in advance of deadlines, confirming the instructors' observations about high levels of interest in game-based activities.

6.4 Flowchart Challenges

The instructors reported that the use of flowcharts as a problem solving tool was the most challenging topic in the program. Unlike Scratch, it was observed that a significant number of the flowchart-related submissions were not timely and/or inaccurately portrayed solutions to problems:

> *...at first we thought it [flowcharts] would have been simple, but it turned out to be extremely complex for the kids. (P1)*

While it is acknowledged that game-based activities were not utilized in this part of the program, additional support was provided through breakout room activities, after-class remediation and peer support; however, the issue persisted. Similar studies have observed this waning interest in flowchart-related activities (Erol & Kurt, 2017) and potential complexity of the topic for younger students (Ali & Saltan, 2015). While further investigation into the student perspective is needed, it may be reasoned that the shift from game-based activities to flowcharts may have also reduced student interest and motivation in the topic area.

The use of flowcharts during Scratch activities may have contributed to further challenges. For instance, while Scratch code blocks such as 'repeat until' hide an iteration's conditional check, the decision component of the flowchart requires it to be explicitly captured. Disconnects of this nature may have limited the students' ability to translate flowchart components to the programming concepts learnt in Scratch. Future research may therefore find it worthwhile to investigate the extent to which flowcharts serve as a compatible problem-solving tool for Scratch.

6.5 Final Scratch Project

It is noted that students were able to exercise 'ownership', a strategic component of the program's teaching model (see Figure 1), in the final assessment. Through individual Scratch projects, they were tasked with proposing their own ideas which they were then expected to design and implement. The instructors largely considered this strategy to be effective since a significant number of students successfully completed the activity, with algorithms and pseudocode being popular choices for problem solving. However, the instructors also highlighted a few instances whereby the students implemented projects that differed from their proposed ideas:

> *...the implementation [final project] in Scratch did not reflect the problem. A few of them... were implementing projects that they didn't even [propose]. (P1)*

While there is a need to investigate this further from the students' perspective, it signalled issues with their willingness to persist with problem solving and seeing a project to its completion. Future virtual iterations of the program may therefore find value in utilizing scaffolded projects via milestones (e.g., Krug et al., 2021). Furthermore, similar studies have found collaborative work to be a strength, due to the positive motivational impact and support that students can provide to each other (Sentance & Csizmadia, 2017).

6.6 Virtual Engagement

Some engagement strategies that were found to be effective in the virtual setting were breakout room activities, live demonstrations of coding, and collaborative debugging and troubleshooting.

Breakout rooms, not greater than ten (10) students, were observed to boost learner-to-instructor and learner-to-learner engagement. For example, when problem solving was taught, the students and instructors were placed into breakout rooms to engage in discussions, collaborate and showcase different ways of solving the same problems. Remarkably, students who contributed less frequently in larger group sessions were more outspoken in the breakout rooms.

Live coding demonstrations also promoted learner-to-instructor engagement and were especially effective for exhibiting samples of model programs. This helped students to better understand what was expected. Screen sharing by students also fostered learner-to-learner and learner-to-instructor engagement by allowing the class to collaboratively debug programs and troubleshoot the Scratch environment.

> *...rather than telling them what to do, we literally demoed it... you have to do this like this, click that, you know... to help them overcome these hurdles. (P2)*

6.7 Virtual Learning Environment Constraints

Constraints observed in the virtual learning environment included issues with communication via Zoom and limited opportunities for social applications of ICT.

The instructors revealed that managing communication during Zoom sessions while simultaneously teaching and sharing their screens was challenging. For instance, if several students interacted via the Zoom chat feature, the instructors would find it difficult to maintain a high level of responsiveness while conducting the lesson. As such, it was recommended that future iterations place more emphasis on verbal communication via Zoom and designate an instructor to monitor the chat.

While session rules such as 'raise hand' and 'wait your turn' were initially put in place to enforce order and efficient communication, the instructors observed that it did not create a comfortable environment for the students, thus:

> *...we say, just a second OK, we're coming to you... in a way to encourage them and keep that momentum going. (P2)*

As a further constraint, in the virtual mode, the range of ways in which social applications of ICT could have been demonstrated was limited. Unlike previous face-to-face iterations of the GGC program, which provided students with first-hand exposure to the field and a real-world appreciation for its application, the virtual mode relied heavily on analogies and explanations for this purpose. As such, future iterations may explore the inclusion of ICT webinars, virtual STEM fairs, virtual reality tours, and live streams.

7. Conclusion

In this study, we explored the adoption of the GGC program's teaching model to the virtual mode of operations. We found that the instructors proactively extended their teaching role to include mentorship. They also encouraged informal learner-to-learner interaction. Notably, these developments are capable of contributing to the creation of a virtual learning community. In addition to this, it was found that Scratch and game-based activities, live demonstrations, breakout rooms and projects that fostered ownership were effective in delivering the program virtually, as opposed to Zoom chat and session rules.

It was also noted that other constraints of the virtual mode, compounded by the COVID-19 pandemic, limited the range of ways in which the social applications of ICT could have been demonstrated. It is therefore recommended that future virtual programs explore alternative techniques.

In addition, parent-driven enrolment may have been a possible cause for challenges in sustaining engagement and motivation. This suggests that there is a need for closer examination of the dynamics of parental involvement, as well as a review of the program's recruitment strategies.

Furthermore, the use of flowcharts for problem solving was observed to be particularly challenging for the students. While we have found studies reporting waning interest and potential complexity with flowchart-related activities, there appears to be limited research on the extent to which flowcharts serve as a compatible problem-solving tool in programs of this nature.

Overall, we recognize that these findings can guide the extension of the GGC program's teaching model, as well as the design and implementation of other ICT training interventions.

References

Abonyi-Tóth, A., & Pluhár, Z. (2019). Wandering Micro: bits in the Public Education of Hungary. Informatics in Schools. New Ideas in School Informatics. ISSEP 2019. Lecture Notes in Computer Science, 11913. Springer, Cham. https://doi.org/10.1007/978-3-030-33759-9_15.

Aivaloglou, E., & Hermans, F. (2019). How is programming taught in code clubs? Exploring the experiences and gender perceptions of code club teachers. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research (pp. 1-10). https://doi.org/10.1145/3364510.3364514.

Ali, O. L. U. K., & Saltan, F. (2015). Effects of using the scratch program in 6th grade information technologies courses on algorithm development and problem solving skills. Participatory educational research, 2(5), 10-20. https://doi.org/10.17275/per.15.spi.2.2.

Alsheaibi, A., Huggard, M., & Strong, G. (2020, October). Teaching within the CoderDojo Movement: An Exploration of Mentors' Teaching Practices. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-5). IEEE. https://doi.org/10.1109/FIE44824.2020.9273998.

Benvenuti, M., Freina, L., Chioccariello, A., & Panesi, S. (2021). Online Scratch Programming With Compulsory School Children During COVID-19 Lockdown: An Italian Case Study. In Handbook of Research on Lessons Learned From Transitioning to Virtual Classrooms During a Pandemic (pp. 167-186). IGI Global. https://doi.org/10.4018/978-1-7998-6557-5.ch009

Bolliger, D. U., & Martin, F. (2018). Instructor and student perceptions of online student engagement strategies. Distance Education, 39(4), 568-583. https://doi.org/10.1080/01587919.2018.1520041.

Burke, Q., & Kafai, Y. B. (2010). Programming & storytelling: opportunities for learning about coding & composition. In Proceedings of the 9th international conference on interaction design and children (pp. 348-351). https://doi.org/10.1145/1810543.1810611.

Chandra, S., Chang, A., Day, L., Fazlullah, A., Liu, J., McBride, L., Mudalige, T., & Weiss, D. (2020). Closing the K–12 digital divide in the age of distance learning. Common Sense and Boston Consulting Group. https://www.bbcmag.com/broadband-applications/closing-the-k-ndash-12-digital-divide-in-the-age-of-distance-learning.

Chen, E., Kaczmarek, K., & Ohyama, H. (2020). Student perceptions of distance learning strategies during COVID-19. Journal of Dental Education. https://doi.org/10.1002/jdd.12339.

Creswell, J. W., & Poth, C. N. (2016). Qualitative Inquiry and Research Design: Choosing among five approaches. Sage publications.

DPI. (2019). Guyanese girls soaring high in technology. https://dpi.gov.gy/guyanese-girls-soaring-high-in-technology/.

DPI. (2021a). ICT programme will prepare youth for future. https://dpi.gov.gy/ict-programme-will-prepare-youth-for-future/.

DPI. (2021b). Office of the Prime Minister – Guyanese Girls Code Summer Camp 2021. https://dpi.gov.gy/office-of-the-prime-minister-guyanese-girls-code-summer-camp-2021/.

DPI. (2021c). Prime Minister's office providing ICT training for youth. https://dpi.gov.gy/prime-ministers-office-providing-ict-training-for-youth/.

Eerd, R. & Guo, J. (2020). Jobs will be very different in 10 years. Here's how to prepare. World Economic Forum. https://www.weforum.org/agenda/2020/01/future-of-work.

Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. Computers in Human Behavior, 77, 11-8. https://doi.org/10.1016/j.chb.2017.08.017.

European Commission. (2021). Women in Digital Scoreboard 2021. https://digital-strategy.ec.europa.eu/en/news/women-digital-scoreboard-2021

Fadde, P. J., & Vu, P. (2014). Blended online learning: Benefits, Challenges, and Misconceptions. Online learning: Common misconceptions, benefits and challenges, 33-48.

Garcia-Ruiz, M. A., Alvarez-Cardenas, O., & Iniguez-Carrillo, A. L. (2021, October). Experiences in Developing and Testing BBC Micro: bit Games in a K-12 Coding Club during the COVID-19 Pandemic. In 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall) (pp. 161-164). IEEE. https://doi.org/10.1109/ICISFall51598.2021.9627364

Goode, J. (2008, March). Increasing Diversity in K-12 computer science: Strategies from the field. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (pp. 362-366). https://doi.org/10.1145/1352135.1352259.

ITU. (2021). Digitally empowered Generation Equality: Women, girls and ICT in the context of COVID-19 in selected Western Balkan and Eastern Partnership countries. ITU Publications. https://www.itu.int/dms_pub/itu-d/opb/phcb/D-PHCB-EQUAL.01-2021-PDF-E.pdf

Jungert, T., Levine, S., & Koestner, R. (2020). Examining how parent and teacher enthusiasm influences motivation and achievement in STEM. The Journal of Educational Research, 113(4), 275-282. https://doi.org/10.1080/00220671.2020.1806015.

Kafai, Y., Jayathirtha, G., Shaw, M., & Morales-Navarro, L. (2021). Codequilt: Designing an Hour of Code Activity for Creative and Critical Engagement with Computing. In Interaction Design and Children (pp. 573-576). https://doi.org/10.1145/3459990.3465187.

Kalelioglu, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. Informatics in education, 13(1), 33-50.

Krug, D.L., Bowman, E., Barnett, T., Pollock, L., & Shepherd, D. (2021, March). Code Beats: A Virtual Camp for Middle Schoolers Coding Hip Hop. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (pp. 397-403). https://doi.org/10.1145/3408877.3432424.

Layne, A., DeFreitas, P., Marks, J., & Lackhan, R. (2020, December). Cultivating Positive ICT Perceptions: an application of the MST-tree model to the 'Guyanese Girls Code' Initiative. In 2020 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 934-940). IEEE. https://doi.org/10.1109/CSCI51800.2020.00174.

Li, C. & Lalani, F. (2020). The COVID-19 pandemic has changed education forever. This is how. World Economic Forum. https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/.

Lim, F. P. (2017). An analysis of synchronous and asynchronous communication tools in e-learning. Advanced Science and Technology Letters, 143(46), 230-234.

Lincoln, Y. S., & Guba, E. G. (1985). Naturalistic inquiry. Sage.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). Educational games for teaching computer programming. In Research on e-Learning and ICT in Education (pp. 87-98). Springer. https://doi.org/10.1007/978-1-4614-6501-0_7.

Martin, F., & Parker, M. A. (2014). Use of synchronous virtual classrooms: Why, who, and how. MERLOT Journal of Online Learning and Teaching, 10(2), 192-210. http://jolt.merlot.org/vol10no2/martin_0614.pdf

McDonald, A. & Dillon, L. K. (2021). Virtual Outreach: Lessons from a Coding Club's Response to COVID-19. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, (Virtual Event, New York, USA) (pp. 934-940). ACM. https://doi.org/10.1145/3408877.3432559.

McReynolds, M. R., Termini, C. M., Hinton, A. O., Taylor, B. L., Vue, Z., Huang, S. C., ... & Carter, C. S. (2020). The art of virtual mentoring in the twenty-first century for STEM majors and beyond. Nature Biotechnology, 38(12), 1477-1482. https://doi.org/10.1038/s41587-020-00758-7.

Merriam, S. B. (1988). Case study research in education: A Qualitative Approach. Jossey-Bass.

Micro:bit Research. https://microbit.org/impact/research/.

Miles, M. B., & Huberman, A. M. (1994). Qualitative Data Analysis: An expanded sourcebook. Sage.

Milgram, D. (2011). How to recruit women and girls to the science, technology, engineering, and math (STEM) classroom. Technology and engineering teacher, 71(3), 4.

Misthou, S., Moumoutzis, N., & Loukatos, D. (2021, April). Coding Club: a K-12 good practice for a STEM learning community. In 2021 IEEE Global Engineering Education Conference (EDUCON) (pp. 955-963). IEEE. https://doi.org/10.1109/EDUCON46332.2021.9454039.

OECD. (2018). Indicator B5 Who is expected to graduate from tertiary education?. In Education at a Glance 2018: OECD Indicators, OECD Publishing, Paris. https://doi.org/10.1787/eag-2018-18-en.

Oh, J., Lee, J., & Kim, J. (2013). Development and application of STEAM based education program using scratch: Focus on 6th graders' science in elementary school. In Multimedia and Ubiquitous Engineering: MUE 2013, 493-501. Springer Netherlands. https://doi.org/10.1007/978-94-007-6738-6_60.

Oluk, A., & Korkmaz, Ö. (2016). Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. 8(11), 1-7. https://doi.org/10.5815/ijmecs.2016.11.01.

Robinson, P. E., & Carroll, J. (2017, April). An online learning platform for teaching, learning, and assessment of programming. In 2017 IEEE Global Engineering Education Conference (EDUCON) (pp. 547-556). IEEE. https://doi.org/10.1109/EDUCON.2017.7942900

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies, 22(2), 469-495. https://doi.org/10.1007/s10639-016-9482-0.

Shadding, C. R., Whittington, D., Wallace, L. E., Wandu, W. S., & Wilson, R. K. (2016). Cost-effective recruitment strategies that attract underrepresented minority undergraduates who persist to STEM doctorates. SAGE Open. https://doi.org/10.1177/2158244016657143.

Šimunović, M., & Babarović, T. (2020). The role of parents' beliefs in students' motivation, achievement, and choices in the STEM domain: a review and directions for future research. Social Psychology of Education, 23(3), 701-719.

Šimunović, M., Reić Ercegovac, I., & Burušić, J. (2018). How important is it to my parents? Transmission of STEM academic values: The role of parents' values and practices and children's perceptions of parental influences. International Journal of Science Education, 40(9), 977-995. https://doi.org/10.1080/09500693.2018.1460696.

Skalka, J., Drlík, M., & Obonya, J. (2019, April). Automated assessment in learning and teaching programming languages using virtual learning environment. In 2019 IEEE Global Engineering Education Conference (EDUCON) (pp. 689-697). IEEE. https://doi.org/10.1109/EDUCON.2019.8725127

Skylar, A. A. (2009). A comparison of asynchronous online text-based lectures and synchronous interactive web conferencing lectures. Issues in Teacher education, 18(2), 69-84.

Staubitz, T., Klement, H., Teusner, R., Renz, J., & Meinel, C. (2016, April). CodeOcean-A versatile platform for practical programming exercises in online environments. In 2016 IEEE Global

Engineering Education Conference (EDUCON) (pp. 314-323). IEEE. https://doi.org/10.1109/EDUCON.2016.7474573

Stelter, R. L., Kupersmidt, J. B., & Stump, K. N. (2021). Establishing effective STEM mentoring relationships through mentor training. Annals of the New York Academy of Sciences, 1483(1), 224-243. https://doi.org/10.1111/nyas.14470.

Stoeger, H., Duan, X., Schirner, S., Greindl, T., & Ziegler, A. (2013). The effectiveness of a one-year online mentoring program for girls in STEM. Computers & Education, 69, 408-418. https://doi.org/10.1016/j.compedu.2013.07.032.

Stoeger, H., Hopp, M., & Ziegler, A. (2017). Online mentoring as an extracurricular measure to encourage talented girls in STEM (science, technology, engineering, and mathematics): An empirical study of one-on-one versus group mentoring. Gifted Child Quarterly, 61(3), 239-249. https://doi.org/10.1177/0016986217702215.

Trilling, B., & Fadel, C. (2009). 21st century skills: Learning for life in our times. John Wiley & Sons.

UNESCO. (2017). Cracking the Code: Girls' and Women's Education in Science, Technology, Engineering and Mathematics (STEM). UNESCO: Paris, France: http://unesdoc.unesco.org/images/0025/002534/253479e.pdf

UNESCO. (2020). Survey on National Education Responses to COVID-19 School Closures. UNESCO. https://tcg.uis.unesco.org/survey-education-covid-school-closures/.

UNICEF. (2020). Towards an equal future: Reimagining girls' education through STEM. UNICEF. https://www.unicef.org/media/84046/file/Reimagining-girls-education-through-stem-2020.pdf.

Vekiri, I. (2010). Boys' and girls' ICT beliefs: Do teachers matter?. Computers & Education, 55(1), 16-23. https://doi.org/10.1016/j.compedu.2009.11.013.

Voogt, J., & Roblin, N. (2012). A comparative analysis of international frameworks for 21st century competences: Implications for national curriculum policies, Journal of Curriculum Studies, 44(3),.299–321. https://doi.org/10.1080/00220272.2012.668938.

Wang, J., & Moghadam, S.H. (2017, March). Diversity barriers in K-12 computer science education: Structural and social. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 615-620). https://doi.org/10.1145/3017680.3017734.

Yamagata-Lynch, L. C. (2014). Blending online asynchronous and synchronous learning. International Review of Research in Open and Distributed Learning, 15(2), 189-212. https://doi.org/10.19173/irrodl.v15i2.1778.

Appendix A

Key components that guided the creation of the focus group questions

The focus group discussion was guided by semi-structured questions related to the following components of the GGC teaching model (see Section 4.1):

3 Motivational Components - Reflection on sources of motivation for the students (e.g., parental involvement, instructor perception, female role models, social applications, etc.).

4 Supporting Components - Investigation of the supporting systems and resources that were made available to the instructors.

5 Engagement and Teaching Strategies - Reflection on the strategic engagement activities that were employed and how they may have facilitated a conducive learning environment for the students (e.g., opportunities to collaborate, communicate and exercise critical thinking and problem solving skills).

6 Challenges - Reflection on specific challenges faced by instructors and students during the programme (e.g., challenging topics, remedial steps, etc.).

Table A1. Focus group questions related to Motivation

MST-tree Model Component	Focus Group Questions
Motivation Components: parents, teachers' perception, female role models, social applications	○ What practices (strategies, techniques) were employed (if any) to motivate and inspire the students? ○ Probing mentor relatability as a motivation strategy: ▪ What techniques did you use to make the girls see you as relatable?

	○ To what extent were parents/guardians involved in the GGC program? ○ Probing the influence of parents'/guardians' involvement on student enrolment and interest ▪ Do you think the parents were more interested in the program than the girls? ○ To what extent did mentors' perception (open opinions, judgements, thoughts, recognition) play a role (e.g. communicating confidence in the students' ability) in the GGC program? ○ To what extent were female role models (e.g. women in ICT - Guyana/Internationally) involved/included in the GGC program? How? ○ To what extent did social applications of ICT (e.g. videos or discussions about real world ICT interventions, field trips, games) play a role in the GGC program?

Table A2. Focus group questions related to Support

MST-tree Model Component		Focus Group Questions
Supporting Components: OPM, Training, UG	**1**	To what extent did OPM play a significant role (e.g. finances, Zoom license, recording videos, Zoom technical support) in supporting the GGC program?
	2	To what extent did the UG (CS Department staff) support (e.g. previous learning materials, curriculum, weekly meetings and guidance) the GGC program?
	3	To what extent did previous training/experiences (e.g. BSc degree program, club involvement, MoPT, STEM Guyana, running UG tutorials) prepare you for the GGC program?

	4 What resources (e.g. Zoom, internet, Google classroom) were made available to you (or utilized) to deliver the program online?
	5 How reliable was internet access for you?
	6 What type of device(s) did you use to conduct the program?
	7 How conducive (comfortable, noise-free, cool, professional) to teaching was the environment in which you conducted the program?

Table A3. Focus group questions related to Engagement

MST-tree Model Component	Focus Group Questions
Engagement/Teaching Components: Strategic Activities, Conducive environment, Problem solving, Critical thinking, Collaboration, Communication, Creativity, Ownership	**8** What engagement strategies did the students respond more positively towards?
	9 What engagement strategy or strategies were least effective?
	10 Did you employ strategic activities (e.g. think-pair-share, plugged, unplugged, whole group, breakout rooms) to explain particular programming concepts? If yes, please provide a few examples.
	10.1 Probing specific engagement strategies:
	10.1.1 How were breakout rooms used?
	10.1.2 How were games used?
	10.1.3 Do you think that the strategic activities had an impact on learning programming concepts?
	11 In what ways did you strive to make the online space a conducive environment (e.g. comfortable, psychologically safe space) for learning?

12 Did any challenge (e.g. issues completing and/or submitting home-work activities, quiet review of concepts not making sense) result in further interactions outside of the weekly sessions?

13 Were the students afforded opportunities to collaborate with each other? If yes, please provide a few examples.

13.1 Probing learner-to-learner collaboration (out of class)

13.1.1 How was the communication group initiated and used?

14 Were the students afforded opportunities to exercise and/or develop their communication skills? If yes, please provide a few examples.

15 Were the students afforded opportunities to exercise and/or develop their creative, problem solving, critical thinking skills? If yes, please provide a few examples.

15.1 Probing the strategies used for problem solving

15.1.1 Given the challenges with flowcharts, how did the students complete problem solving activities?

16 Were the students afforded opportunities to exercise and/or develop a sense of ownership (e.g., freedom to create and share individual ideas)? If yes, please provide a few examples.

16.1 Probing opportunities for ownership

16.1.1 How would you compare activities that encourage creativity and ownership against those that are more structured?

Table A4. Focus group questions related to Challenges

MST-tree Model (Overview)	Focus Group Questions
Challenges	**1** What difficulties, if any, have you experienced teaching programming (e.g. of barriers - explaining concepts, teaching or visual aids) in the online mode?
	2 What topics in the GGC curriculum were most challenging for the students? Why?
	2.1 Probing challenges with flowcharting
	2.1.1 Why do you think flowcharting was so problematic for the students?
	3 Do you think the virtual mode made it harder to problem solve with flowcharts?
	4 What topics in the GGC curriculum were most challenging (e.g. time and effort to prepare, uncertainty, a struggle to engage the class) to teach? Why?
	5 Were remedial steps (e.g. teaching concepts using a different method to improve understanding) taken to help students to understand topics that they found more challenging? If yes, please provide details regarding the remedial steps that were taken and what was the outcome.

The Effect of the Modality on Students' Computational Thinking, Programming Attitude, and Programming Achievement

Ibrahim Cetin[1]

Tarik Otu[1]

[1]Computer Education and Instructional Technology Department, Education Faculty, Abant Izzet Baysal University, Bolu, Turkey

[2]Orhangazi Middle School, Bolu, Turkey

DOI: https://doi.org/10.21585/ijcses.v6i2.170

Abstract

The purpose of the current study was to explore the effect of modality (constructionist mBlock, Scratch, and Python interventions) on six-grade students' computational thinking, programming attitude, and achievement. The pre-test and post-test quasi-experimental design was used to explore the research questions. The study group consisted of 105 six grade students from three different classes. A constructionist learning environment was formed for Scratch, mBlock, and Python groups. All groups were given 8 week-instruction. Instruction included two forty-minute sessions each week. The data were collected through the programming achievement test, computational thinking test, and computer programming attitude scale. The results of the study showed that mBlock group outperformed the Scratch and Python groups with respect to computer programming attitude. Students who attended mBlock and Scratch groups had higher levels of programming achievement than those of the students who attended the Python group. No significant differences with respect to computational thinking were observed between the groups. This study has implications for educators who are teaching computational thinking and programming. Further research was recommended to explore the effect of modality.

Key Words: Modality; computational thinking; programming; constructionism

1. Introduction

Computational thinking and programming have become important skills for educators around the world. Researchers are searching for the best practices to help students improve their computational thinking and programming (Tikva and Tambouris, 2021). Educational institutions are adapting computational thinking and programming concepts into their regular curriculums. It is contended that computational thinking is related to problem-solving, abstraction, critical thinking, and creativity (Korkmaz, Cakir, and Ozden, 2017; Cakiroglu, Cevik, Koseli, and Aydin, 2021; Panskyi,

Rowinska,and Biedron, 2019). Nevertheless, computational thinking is a relatively new area in educational research. There is no commonly agreed-upon definition of computational thinking in the literature yet. Researchers have proposed different definitions for computational thinking.

The term was first used by Seymour Papert (1980). Papert used the term computational thinking without providing a definition. He considered computational thinking in the context of the educational theory called constructionism which is a reconstructed form of Paiget's constructivism (Papert, 1993). Papert suggested that computational thinking can be used to help students improve their mathematical knowledge. He considered programming as a medium to construct a relatively concrete product. Students can think on a relatively concrete product to improve abstract mathematical knowledge.

Ed Dubinsky (1995), like Papert, contended that formal mathematical thought should be grounded in experience. Dubinsky and his colleagues reconstructed Piaget's constructivist theory of learning in the context of collegiate mathematics education (Arnon et al., 2013). They constructed APOS theory. They did not use the term computational thinking, but they stressed the power of computing in mathematics education. The core of APOS theory is related to reflective abstraction. Dubinsky considered programming as a unique tool to help students construct necessary mathematical abstractions. Moreover, it was contended that the nature of abstraction in mathematics is the same in computational thinking (Cetin and Dubinsky, 2017). Papert and Dubinsky are important researchers in mathematics education in that they both considered computational thinking from their systematic learning theory perspectives.

Before educational theorists, computer scientists emphasized the terms algorithm and algorithmic thinking. The term algorithmic thinking was used by the researchers before computational thinking to express the core of computer science (Denning, 2017). Knuth (1985, p.172) contended that "… Computer Science is the study of algorithms" and he stressed the importance of algorithmic thinking in the context of computer science. Newell, Perlis, and Simon (1967) took a different perspective and proposed that computers are not only tools, but there are also phenomena surrounding computers. Computer science deals with phenomena and algorithms and the hardware is the important element of the phenomena.

Computational thinking was first defined by Wing (2006) as the application of computer science concepts to solve problems design systems and understand human behavior. Aho (2012, p.832) emphasized the role of the computational model in computational thinking and defined computational thinking "…to be the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms". Cuny, Snider, and Wing (2010) modified Wing's early definition and considered computational thinking as "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (as cited in Wing, 2011; p. 20). This

definition is taken as a base for this study. Brennan and Resnick (2012), depending on their Scratch teaching experience, developed a computational thinking framework including computational concepts (programming concepts), computational practices (practices used in problem-solving), and computational perspectives (the self-reflection on computing practices).

The theoretical and historical perspectives of computational thinking are important. Moreover, an instructional perspective that provides means to help students improve their computational thinking and programming skills is important for educators. Programming and computational thinking are not easy for students to comprehend. There are studies that report students have trouble in learning programming (Chao, 2016; Moons and Backer, 2013; Sáez-López, Román-González, and Vázquez-Cano, 2016). When beginners learn programming and computational thinking there are possible pathways for them to follow. They can be introduced to computational thinking or programming with the help of block-based programming, robotics programming, text-based programming, and computer science unplugged approach. Educators can also use a blended approach by mixing some of these ways. Nevertheless, there are not enough guidelines to pick one of the modalities to help students learn computational thinking and programming. This study will explore the effect of modality (block-based, text-based, and robotics) in the context of six-grade students to start filling this gap.

1.1 Literature Review

Pioneer computer scientist Dijkstra (1982) stated that "The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities" (p. 129). However, how the tool will be used is not self-evident in the tool itself. Herrmann (2003) stated that "A technical system is not a product of its own but is made and controlled from outside… Technical systems serve purposes which do not lie within themselves but are assigned from other systems" (p. 62). Therefore, the pedagogical approach behind the tool or modality should be shortly explained before giving the details of the related literature about modalities. In the current study, constructionism will be utilized as a pedagogical approach. Constructionism was constructed on the theory of Piaget's constructivism. Constructivism is related to the origins and development of knowledge (Piaget, 1964). In the learning process, an individual acts on an internal/external object, transforms it, constructs new knowledge at a higher level of plane, and integrates the new knowledge with the existing ones at the higher level of plane. The mental mechanism is the reflective abstraction in the development of logico-mathematical knowledge. The mental structure developed through reflective abstractions is called schema. Schema is a more or less coherent collection of mental structures. Individuals actively construct their own knowledge or mental structures. Papert (1980), the constructor of the constructionist approach, agrees with Piaget's theory of knowledge construction. He furthers it with the idea of the development of a concrete entity. Abstract concepts can be represented as computer

programs or algorithms. These programs are concrete in the sense that their results, e.g., moving turtle, robotic tasks, or games, can be seen immediately after running them. Therefore, individuals can deal with abstract concepts through concrete means. Individuals can share their entities/programs with others to collectively think on it. The computer becomes a cultural tool that supports individuals' learning as in the case of the support for the learning mother tongue.

Constructionism is the pedagogical approach behind all the modalities in the current study. The modalities are constiructionist Scratch, Python, and mBlock learning environments. There are many block-based programming environments available for instructional purposes. Scratch, Alice, and App Inventor are the ones that are used frequently (Hu, Chen, and Su, 2021). When beginners start learning to program with text-based programming tools they need to handle complicated syntax issues. They need to memorize programming statements, write codes in a correct way and debug the program when needed. In contrast to text-based programming environments, block-based programming environments are intuitive (Xu, Ritzhaupt, Tian, and Umapathy, 2019). Beginners can construct games, animations, and mobile applications by using block-based environments. These are complex programming products that are hard to develop with text-based languages. Block-based programming environments can provide beginners with concrete and joyful experiences (Topalli and Cagiltay, 2018). Beginners can run the script and see the result on the screen. They can develop programs that are interesting for themselves (Mladenović, Mladenović, and Žanko, 2020). Resnick et al. (2009) summarized the expected features of block-based programming environments for beginners as low floor (easy to get start), high ceiling (allows beginners to construct complex projects), and wide walls (supports the development of meaningful products).

Programs are represented as plain text in text-based programming. There is a variety of paradigms in text-based programming and text-based programming is the norm in the industry (Kandemir, Kalelioğlu, and Gülbahar, 2021). Python, Java, and C++ can be given as examples of text-based programming languages. There is a belief that text-based programming is harder for beginners (Kölling, 2015). Nevertheless, this does not directly mean that text-based programming should not be used for the first programming experience. Some arguments support the use of text-based programming and discourage block-based programming environments for beginners. Mihci and Donmez (2017) contended that some university students are not interested in block-based programming environments. They chose text-based programming environments since they believe that text-based programming environments are the industry standard that might help them for their future career. There are several approaches for educational text-based programming that aims to introduce students in a more beginner-friendly way. These approaches are mini-language, sub-language, visualization, and frame-based programming (Brusilovsky et al., 1997; Cetin, 2020; Kandemir, Kalelioğlu, and Gülbahar, 2021; Kölling, Brown, and Altadmri, 2017). In the current study, the mini-languages approach will be focused on since it was used in one of the interventions in the study. In

mini languages, there is an actor (e.g., turtle or robot) in a microworld. Students control the actor by using commands of the mini language. The mini language generally includes simple commands and basic programming structures. Students can see the immediate result of their program via the actor in the microworld. Mini languages have current implementations in programming education. For example, Python has a turtle library which is a relative of Logo turtle. Students can use this library to get the basics of Python and experience programming with the mini-language approach.

Educational robotics have a place in computational thinking and programming education. There is a variety of robotic kits that can be used for this purpose. Some of these tools are Lego robots, Bee-Bot, MBot, and Arduino kits. Beginners can use a prebuild robot (Bee-Bot); they can build a robot by a using controller unit, motors, sensors, cables, and technic elements (Lego Mindstorms EV3 and MBot); or they can build a robot by using a microcontroller, basic electronic elements, modules, sensors, motors, cables, and mechanic elements (Arduino kits). Robotic kits can be programmed by using text-based (MicroPython and Arduino IDE), block-based (Scratch, mBlock, ArduinoBlocks), hybrid (RobotC) programming environments/libraries, and by just pushing buttons on the kit. Programs for robotic kits can be written on computers just as in the case of block-based and text-based programming, then the program can be downloaded to the robot. After the program is downloaded to the physical robot, the robot can get data from the physical world, interpret it through its microcontroller, and downloaded the program; hence the robot creates a reaction through its actuators. In this way, the program in the computer gets a connection with the physical world. Moreover, data from the physical world can be transferred to the computer through robotic kits to form an interaction between the physical world and computers. The two-way physical world and computer connection can provide more meaningful activities for students (Sullivan and Bers, 2016).

Although literature reviews and meta-analysis studies (Hu, Chen, and Su, 2021; Noone and Mooney, 2018; Xu, Ritzhaupt, Tian and Umapathy, 2019) related to block-based versus text-based programming provide promising results, they are mainly inconclusive. Noone and Mooney (2018) conducted a systematic review study including 29 studies published in journals and conference proceedings. They proposed that block-based programming provides benefits over text-based programming. Xu et al. (2019) conducted a meta-analysis study to compare the effect of block-based and text-based programming environments on novice students' cognitive and affective scores. They compiled 13 studies published in journals and conference proceedings. They contended that there is a small effect size in favor of block-based programming environments with respect to cognitive scores. The overall effect size was not found to be significant. Considering the education level, the effect size in the middle school context was the smallest. Moreover, they stated that there is a trivial effect size with respect to affective scores. The effect size for affective scores in the middle school level was insignificant and the overall effect size was also insignificant. Hu et al. (2021) conducted a meta-analysis study to explore the effect of block-based programming on students' academic achievement.

They examined 29 empirical studies published in journals and conference proceedings. They reported an overall small to medium significant effect size in favor of block-based programming. Educational level was found to be a moderator variable. A large effect size was found for elementary and middle school students.

The situation is better in the robotics programming context. The literature review and meta-analysis studies mainly reported positive results in favor of robotics programming for teaching programming and computational thinking. Major, Kyriacou, and Brereton (2012) conducted a systematic literature review to explore the effect of using robots in teaching novices programming. The languages used for programming robots were mostly text-based languages (e.g., Java, C++, and Ada). They considered 23 studies for physical robot programming; (i) 16 of the 23 studies found educational robotic effective for introductory programming instruction; (ii) four of the studies had mixed results; (iii) one study was classified as ineffective; and (iv) two studies were unclassifiable. Scherer, Siddiq, and Viveros (2016) conducted a meta-analysis to consider the effectiveness of block-based programming and educational robotics. They examined 20 studies for the block-based programming condition and 7 studies for the educational robotics condition. They concluded a significantly moderate effect size in favor of block-based programming and a significantly large effect size for educational robotics. Zhang, Luo, Zhu, and Yin (2021) explored the effectiveness of educational robotics. They had considered 17 studies in the meta-analysis. They found a significant moderate effect size in favor of educational robotics with respect to computational thinking.

As seen in the literature, some studies propose that a kind of programming environment or modality has the potential to promote better learning outcomes (Weintrop and Wilensky, 2017). Some studies propose the reverse; similar tools do not result in better learning outcomes (Mihci and Donmez, 2017). Most of these studies were done in the context of programming education. The computational thinking perspective has not been given enough attention yet. Moreover, cognitive variables were the main focus in most of these studies. There is a limited number of studies related to affective variables like attitude. Therefore, there is no consensus in the literature related to effectiveness of robotics, block-based and text-based programming environments. Beside this, when the first text-based programming course should be given is another issue: is middle school context suitable for text-based programming, and if yes what is the optimum grade to start text-based programming? The aim of this study is to compare the effect of constructionist learning instruction that was given in robotics, block-based, and text-based contexts on sixth-grade students' programming achievement, computational thinking, and attitudes towards computer programming. mBlock (with MBot) was used for the robotics context; Scratch was used for the block-based context; and Python with turtle library was used for the text-based context. The followings are the research questions to be explored in the current study.

1. Is there any significant mean difference between the groups that were given mBlock (with MBot), Scratch, and Python (with turtle library) based constructionist instruction with respect to students' programming achievement?
2. Is there any significant mean difference between the groups that were given mBlock (with MBot), Scratch, and Python (with turtle library) based constructionist instruction with respect to students' post-computational thinking scores when their pre-computational thinking scores were controlled?
3. Is there any significant mean difference between the groups that were given mBlock (with MBot), Scratch, and Python (with turtle library) based constructionist instruction with respect to students' post-computer programming attitude scores when their pre-computer programming attitude scores were controlled?

2. Material and Methods

The current study utilized a quasi-experimental design with three sixth-grade introductory programming classes. There were six sixth-grade classes in the school in which the study was carried out. Three study groups were randomly chosen from six classes to construct mBlock (with MBot), Scratch, and Python (with turtle library) groups. In this study, for the simplicity mBlock (with MBot) group will be called mBlock and Python (with turtle library) group will be called the Python group. Before the intervention, all three groups were given a computational thinking test (CTT) and computer programming attitude scale for middle school students (CPAS-M) as pre-tests. After conducting pre-tests, the eight-week intervention period had started. Students were given two 40-minute sessions each week. The interventions in all three groups were designed based on a constructionist approach. The difference among the groups was the programming environment. After the intervention period, all three groups were given CTT, CPAS-M, and a programming achievement test (PAT). The design of the study is summarized in Table 1.

Table 1. The Design of The Study

Group	Pre-test	Programming Env.	Post-test
MBlock	CTT	MBlock with MBot	CTT
	CPAS-M		CPAS-M
			PAT
Scratch	CTT	Scratch	CTT
	CPAS-M		CPAS-M
			PAT
Python	CTT	Python with turtle	CTT
	CPAS-M	library	CPAS-M
			PAT

2.1 Subject

105 sixth-grade students from three intact classes were included in the current study. 45 of the students were female and 60 of them were male. There were six sixth-grade classes in the school in which the study was conducted. mBlock, Scratch, and Python groups were randomly chosen from the six classes. The students were given an information technology and software course as their regular curriculum. This course generally starts at fifth grade in the country. Nevertheless, schools that have extensive English language teaching for fifth graders, provide the course at the sixth-grade level. For the study school, the information technology and software course were first given in the sixth grade since there was an English language teaching program for the fifth graders. The given course was compulsory and students' first programming course in their formal education. The mBlock group consisted of 36 students (16 females and 20 males); the Scratch group consisted of 34 students (14 females and 20 males); and the Python group consisted of 35 students (15 females and 20 males).

2.2 Intervention

The same instructor instructed in all three groups. The same approach was used in three groups. The instructions in three groups were designed based on constructionism and pair programming. The programming environments were different in the groups. mblock with Mbot was used in the mBlock group; Scratch was used in the Scratch group; turtle library of Python was used in the Python group. The students studied in pairs in the computer laboratory. Interventions lasted eight weeks, two 40-minute sessions each week. The instruction aims to improve students' computational thinking and problem-solving skills by using constructs of programming with a programming language. The main objectives of the instruction were:

i. Design algorithms,
ii. Know and use programming structures (e.g., variables, conditionals, loops, and functions),
iii. Solve problems by using programming structures,
iv. Choose and apply appropriate programming approaches to solve problems,

The instructions in three groups can be designed in such a way that they all include similar activities. The activities can be given in similar sequence. We believe that this is not a good way to compare the effectiveness of the programming environments. Each programming environment has different potential. The advantages that each modality brings to computer science education is different. Designed instructions should consider peculiarities of programming environments. Therefore, the instructions in the study were designed considering the peculiarities of programming environments. Nevertheless, this does not change main objectives of the instructions. The objectives are the same, e.g., design algorithms and use appropriate programming structures to solve problems. Our approach is different ways with their peculiarities to same ends.

The following is an activity from the mBlock group. At the beginning of the activity, students were shown the line follower robot in Figure 1. They were asked to write a mBlock code that makes the robot follow the black strip shown in Figure 1.

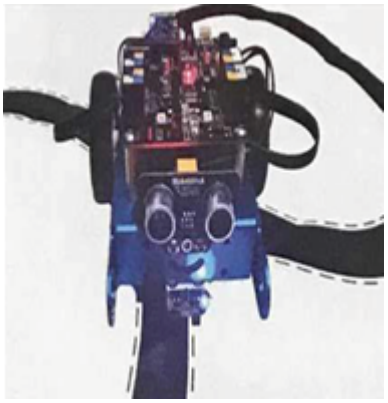


Figure 1. Example mBlock Activity

The following is an activity from the Scratch group. At the beginning of the activity, students were shown the screen in Figure 2. They were asked to create a game similar to the one shown in Figure 2. The bowl in the game can be controlled on the x-direction with the keyboard or the mouse. Apples spawn and fall down from random positions at the top. The player tries to take the apples. Each apple provides a constant point. There is a time limit in which the player tries to get the highest possible score.

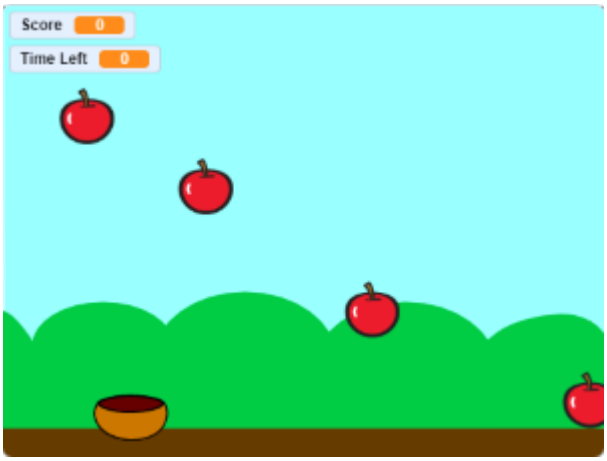


Figure 2. Example Scratch Activity

The following is an activity from the Python group. At the beginning of the activity, students were shown the shape in Figure 3. They were asked to write Python code that draws a shape similar to the one shown in Figure 3.

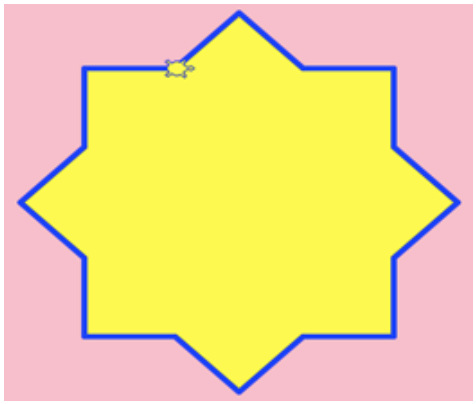


Figure 3. Example Python Activity

The instructor had the role of guide to help students explore. Students were given tasks in the laboratory. They studied to complete the given tasks. The tasks were mainly in one of the following six forms:

i. The instructor gave certain codes and asked students to find the function of these codes,
ii. The instructor gave a task and asked students to discuss as a class how to handle the given task,
iii. The instructor pointed out certain codes, and asked students to complete a given task,
iv. The instructor gave a task and asked students to complete it without any cues,
v. The instructor asked for a class discussion when a common conceptual issue appeared,
vi. The instructor asked students to develop their products.

The instructor tried to help students explore programming concepts through these tasks. The instructor tried not to give complete answers to the students. Whenever necessary, the instructor explained the code and how he handles problems at the hand. But this was kept minimum. Students were encouraged to find their own ways. The instructor was present in all laboratories. The instructor monitored group and individual work and gave group and individual feedback.

Students studied in pairs in the laboratories. Each pair had one computer in all classes. Additionally, each group had one MBot kit in the mBlock group. There were two roles in pairs. One of the students had the keyboard and mouse and was responsible for code writing. The other student reviewed the code writing process; monitored the problems; and tried to help handle the task at the hand. The pairs continuously changed their roles from task to task. Intra-group communication was allowed in the laboratories. Pairs discussed the issue whenever necessary. They were warned not to give complete answers but to negotiate their ideas.

2.3 Instruments

The study included three instruments to gather data. These instruments were a computational thinking test (CTT), a computer programming attitude scale for middle school students (CPAS-M), and a programming achievement test (PAT). CTT and CPAS-M were both used as pre and post-tests. PAT was used for the post-test.

PAT was developed by the authors of this study. PAT has three versions for mBlock, Scratch, and Python groups. All three versions included identical items. 25 items were developed considering the aims of the information technology and software course by the authors of this study. 15 of the items were selected based on content validity. The items were given to two domain experts and two language experts. Domain experts evaluated the items in terms of content validity and appropriateness for students' grade level. Language experts evaluated the items in terms of comprehensibility and grammatical aspects. Then all three versions of the PAT were sent to two domain experts. They evaluated each item in three versions and checked if the items are identical or not. They scored each item from 0 to 10. 0 means "the questions in the three versions are completely different" and 10 means "the questions in the three versions are exactly the same". Moreover, they provided feedback if the item was not given 10. One item had 8, and one item had 9 points. The scores of the remaining items were 10. Necessary changes were done depending on the feedback. Then the Scratch version of PAT was applied to 169 (73 females and 96 males) seventh graders who had already taken an information technology and software course and used Scratch in their classes. The gathered data was analyzed by using TAP software. Since one of the items had improper difficulty (0.11) and discrimination (0.05) values, it was removed from the PAT. Item difficulty and discrimination values of PAT are summarized in Table 2.

Table 2. Item statistics for PAT

Item #	Item Difficulty	Item Discrimination
1	0,86	0,31
2	0,86	0,36
3	0,26	0,49
4	0,62	0,65
5	0,44	0,68
6	0,57	0,50
7	0,38	0,73
8	0,47	0,43
9	0,84	0,36
10	0,86	0,34

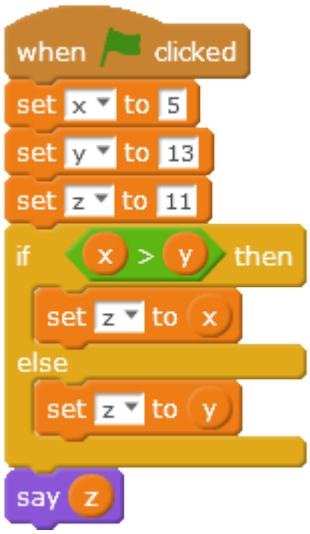
11	0,70	0,38
12	0,73	0,62
13	0,41	0,45
14	0,56	0,47

The mean item difficulty of the PAT was found to be 0.61 and the mean item discrimination of the PAT was 0.48. The internal consistency coefficient (KR20) was found to be 0.74 for the 14-item test. The following three questions can be given as an example for versions of the same question for the three groups.

Example Question for the Three Groups

mBlock Group What is the output of the following code?

a. 5
b. 13
c. 11
d. 18



Scratch Group What is the output of the following code?

a. 5
b. 13
c. 11
d. 18

Python Group What is the output of the following code?

a. 5
b. 13
c. 11
d. 18

```
x=5
y=13
z=11
if x>y :
        z=x
else :
        z=y
print(z)
```

The CTT was originally developed by Román-González, Pérez-González and Jiménez-Fernández (2017) in Spanish. CTT was adapted to Turkish. The test aims to assess middle school students' computational thinking levels. It is a multiple-choice test and includes 24 items related to computational concepts. Each item has four choices. KR 20 value of the test was reported 0.78 in the adaptation study. KR20 value was found 0.76 in the pre-test and 0.79 in the post-test in the current study. The total score can a student get from the CTT ranges from 0 to 24.

The CPAS-M was constructed by Gul, Cetin, and Ozden (2022). It was developed to assess middle school students' attitudes towards programming. It includes 13 Likert-type items. The maximum score that a student can get from CPAS-M is 65 and the minimum score is 13. The scale is one-dimensional. Cronbach alpha coefficient of the original scale was found to be 0.93. In the current study, the Cronbach alpha coefficient was found to be 0.91 and 0.93 correspondingly for pre-test and post-test.

3. Results

For the first research question, one-way ANOVA was conducted to examine difference(s) between groups in terms of the PAT scores. Descriptive statistics related to PAT scores of groups were given in Table 3. It was observed that Scratch and mBlock groups had close means while the Python group had the lowest mean.

Table 3. Descriptive statistics of PAT

Group	N	M	SD	Skewness	Kurtosis
Scratch	35	54.57	9.58	-0.588	-0.430
mBlock	34	53.09	9.05	0.201	-0.496
Python	31	27.58	12.44	-0.746	0.403

The one-way ANOVA result showed that there was a significant effect of treatment on students' PAT scores at $p<0.05$ level for Scratch, mBlock, and Python groups [$F_{(2-97)}=68.55$, $p<0.05$]. The calculated effect size for this difference was big ($\eta^2=0.59$) (Green and Salkind, 2013). Post hoc comparisons

using the Dunnett C test indicated that the difference was significant between Scratch (M=54.57, SD=9.58) and Python (M=27.58, SD=12.44) groups and mBlock (M=53.09, SD=9.05) and Python (M=27.58, SD=12.44) groups (Table 4).

Table 4. PAT ANOVA Results

Source	SS	df	MS	F	p	Sig. Dif.
Between	14788.145	2	7394.072	68.55	0.00	Scratch-Python; mBlock-Python
Within	10462.855	97	107.864			
Total	25251.000	99				

For the second research question, a one-way ANCOVA was conducted. Before the main analysis one-way ANOVA was conducted to check whether there was a significant mean difference between groups' pre-CTT scores. The ANOVA results showed that there was a significant difference in students' pre-CTT scores at $p<0.05$ level for Scratch, mBlock, and Python groups [$F_{(2-96)}=5.35$, $p<0.05$]. The effect size for this difference was found as medium ($\eta^2=0.10$). Post hoc comparisons using the Tukey test indicated that the difference was significant between Scratch (M=15.17, SD=3.82) and Python (M=11.80, SD=4.80) groups. There was no significant difference between mBlock (M=13.66, SD=3.77) and other groups (Table 5).

Table 5. pre-CTT ANOVA Results

Source	SS	df	MS	F	p	Sig. Dif.
Between	183.636	2	91.818	5.35	0.006	Scratch-Python
Within	1647.536	96	17.162			
Total	1831.172	98				

One-way ANCOVA analysis showed that there was not a significant difference between groups in terms of their post-CTT scores when their pre-CTT scores were controlled [$F_{(2-90)}=0.668$, $p>0.05$]. ANCOVA results are summarized in Table 6.

Table 6. CTT ANCOVA Results

Source	SS	df	MS	F	p
Pre-CTT	636.267	1	636.267	55.917	0.000
Group	15.204	2	7.602	0.668	0.515
Error	1024.088	90	11.379		
Total	1774.809	93			

For the last research question, one-way ANCOVA was utilized. Before the main analysis, one-way ANOVA was conducted to examine whether there was a significant mean difference between groups' pre-CPAS-M scores. The ANOVA results indicated that there was no significant difference in students' pre-CPAS-M scores at $p<0.05$ level for Scratch, mBlock, and Python groups [$F_{(2-93)}=0.783$, $p>0.05$]. The results are summarized in Table 7.

Table 7. pre-CPAS-M ANOVA Results

Source	SS	df	MS	F	P
Between	142.350	2	71.175	0.783	0.460
Within	8452.806	93	90.890		
Total	8595.156	95			

One-Way ANCOVA results showed that a significant difference between groups' adjusted mean CPAS-M scores was observed [$F_{(2-91)}=4.703$, $p<0.05$]. Results were summarized in Table 8. The effect size for this difference was small ($\eta^2=0.094$). Post hoc comparisons using the Bonferroni test indicated that the difference was significant between mBlock (M=51.95) and Python (M=45.28) groups and mBlock (M=51.95) and Scratch (M=46.31) groups. mBlock group significantly outperformed Scratch and Python groups on post-CPAS-M, $p<0.05$.

Table 8. CPAS-M ANCOVA Results

Source	SS	df	MS	F	p
pre-CPAS-M	3384.04	1	3384.04	39.755	0.000
Group	800.586	2	400.293	4.703	0.011
Error	7746.113	91	85.122		
Total	230239.0	95			

4. Discussion and Conclusion

The aim of this study was to assess the impact of modality on sixth-grade students' computational thinking, programming achievement, and programming attitude. mBlock with Mbot, Scratch, and Python with turtle library were used as programming environments. All three groups (mBlock, Scratch, and Python) were given an eight-week intervention, developed considering the principles of constructionism. CTT (computational thinking test) and CPAS-M (programming attitude scale for middle school students) were given as both pre and post-tests. PAT (programming achievement test) was given as a post-test for the groups.

There are studies in the literature contending that robotics and block-based programming provide students with better learning opportunities for programming and programming is one of the best ways

to teach computational thinking (Scherer, Siddiq, and Viveros, 2020; Zhang, Luo, Zhu, and Yin, 2021). So, one might expect that programming instruction with robotics and block-based programming produces significantly better learning outcomes with respect to programming achievement and computational thinking. As expected, mBlock and Scratch groups significantly outperformed the Python group with respect to programming achievement. But there was no significant difference between groups considering students' computational thinking. Moreover, there are studies in the literature contending that robotics and block-based programming provide students with concrete and authentic learning opportunity in which students express themselves better and have joy. So, one might expect that both robotics and block-based programming are better environments related to students' attitudes. However, the Scratch group did not meet expectations. mBlock group significantly outperformed Scratch and Python groups with respect to students' CPAS-M scores.

Considering students programming achievement scores, there was (i) no significant difference between the mBlock and Scratch group, (ii) a significant difference between mBlock and Python groups in favor of the mBlock group, and (iii) a significant difference between Scratch and Python groups in favor of Scratch group. mBlock and Scratch groups were superior to the Python group. This study supports the idea that constructionist block-based and robotics programming environments can provide a better learning experience in terms of students' programming achievement (Kert, Erkoc, and Yeni, 2020). This achievement can be explained by the type of activities that students experienced in their groups. Students in mBlock and Scratch groups constructed physical robots and games/animations respectively. Students in the Python group constructed turtle-based graphics. Although students were able to handle abstract programming concepts through concrete means (programs for the robot, game/animation, and turtle) and see results of their programs immediately in all three groups, robot and game/animation activities might be more engaging. Students can show or tell their acting robots to friends and families, or they can show their games/animations to friends and families and ask them to play their games. Nevertheless, in the case of turtle programming, the graphics on the screen might not be attractive for students themselves and their friends and families. Products of robotic and block-based programming have the potential to be a part of the wider context and to be a cultural tool (Papert, 1993) on which students, peers, friends, teachers, and families can think, talk and give feedback. Therefore, it might be said that robotics and block-based programming environments can provide a rich learning experience for students to achieve in programming since these environments have the potential to be a cultural tool to support students.

There were no significant mean differences between mBlock, Scratch and Python groups with respect to students' post-CTT scores when students' pre-CTT scores were controlled. Two issues should be considered depending on the results related to computational thinking: (i) why there was no significant difference between groups with respect to students' CTT scores and (ii) why this no significant difference phenomenon was observed while there was a significant difference between groups in terms

of students' programming achievement score. One possible explanation might be related to the difference between the nature of achievement and skill. Programming achievement is related to and focuses more on the concepts that are given throughout the course. Computational thinking skill is more general, and it is hard to achieve components of computational thinking like abstraction and algorithmic thinking, problem-solving. Therefore, one can suggest that the first programming course might not be enough to help students improve their computational thinking and short-term intervention might not be representative for the general case. In addition to this, contrary to the common belief, one might contend that the complexities of text-based programming, e.g. syntax and debugging, create opportunities for students to deal with problems. These complexities might provide a learning environment in which students have to deal with problems and improve their computational thinking skills, e.g. problem solving while involved in problem-solving. The syntax of Python is not too complex. The right dosage of syntax and debugging issues might have a positive effect on students' computational thinking. The last explanation of the issues might be that there is no immediate significant relation between programming achievement and computational thinking. There is an approach called CS unplugged that aims to improve computational thinking without using programming. Bell and Vahrenhold (2018) stated that CS unplugged approach is promising for developing students' computational thinking. Therefore, improved achievement in programming might not directly mean improvement in computational thinking.

It was found that there is a significant difference between mBlock, Scratch, and Python groups with respect to students' post-CPAS-M scores when students' pre-CPAS-M scores were controlled. mBlock group significantly outperformed Scratch and Python groups. This result might be related to the interaction of constructionism, students' developmental stage, and properties ofprogramming environments. Six-grade students may not be complete abstract thinkers according to the stage theory of Piaget (Huitt and Hummel, 2003). Students might feel they are not good enough to deal with abstract concepts. Constructionism posits that by developing concrete products, students can deal with abstract concepts through concrete means. This might help students feel better in programming. Among mBlock, Scratch, and Python most concrete form of modality belongs to mBlock. mBlock brings programming into students' daily life. Exploration in the programming instruction as suggested by constructionism happens in the most concrete form in mBlock condition. Students might feel good at programming while producing a product in their physical space. This is related to the perception of students not to their actual achievement. It is possible that the use of robotics with constructionism might help students feel they are good at programming. Nevertheless, this result might simply be due to the novelty effect too. Text-based programming is the oldest way to teach programming and block-based programming is widely used in education in Turkey. Nevertheless, robotics programming is a relatively new approach that is not commonly used in state schools in Turkey yet. Therefore, more interest in the mBlock group might be due to new technology, namely robotics. Krendl and Broihier

(1992) conducted a study to examine the evolution of students' perceptions about computers. They demonstrated strong evidence of novelty effect, particularly in the case of affective responses. In addition to these issues, no significant difference between Scratch and Python groups should be considered. It is contended that students can have more concrete and joyful experiences in block-based programming (Topalli and Cagiltay, 2018). So, one can expect a significant difference with respect to students' CPAS-M scores between Scratch and Python groups in favor of Scratch. We believe that the finding in the current study does not disprove the concrete and joyful experience that block-based programming can provide. It might be the case that Python is perceived as "real" programming that programmers (like game developers and hackers) use. This might affect the perception of students related to programming (Mihci and Donmez, 2017).

Considering the latest literature review and meta-analysis studies related to computational thinking and programming, there is a lack of studies related to the effect of the programming environment on cognitive and especially affective variables. It is a good idea to speculate on the results of such studies from different perspectives until focal points related to the effects of the programming environment are determined by the researchers. The current study utilized this line of reasoning to explain the findings. The results of this study provided possible answers and new questions. The most solid result is that in a constructionist learning environment mBlock is better than Scratch with respect to programming attitude and mBlock is better than Python with respect to programming achievement and attitude in the current situation. Hence practitioners and researchers can use robotics programming to increase the possibility of success of computational thinking and programming instruction for six grade students. Considering the results related to computational thinking, Python seems to be a promising tool. Nevertheless, the Python group failed in programming achievement. It might not be a good idea to use Python as the first programming environment for sixth-graders. Future studies can test the effectiveness of Python for seventh and eighth-graders. Introducing programming with block-based or robotics programming and then utilizing Python might produce effective results. Moreover, there is a newly developing game programming library called Pygame Zero for Python. As its name suggests it is a simplified version of PyGame for educational purposes. Future studies can test its effectiveness of it. In addition to these considerations, researchers and practitioners need to consider the cost. mBlock provides additional costs for students, teachers, and schools. If the cost is not affordable, then Scratch seems to be a good alternative.

There are certain limitations to this study. Firstly, the study was conducted with limited sample size. The generalizability of the findings is limited. It would be better to conduct the study with a larger sample size. Secondly, the study sample is composed of six graders. Six graders cannot be fully counted as abstract thinkers. The cognitive stage is an important factor in education. The findings of this study might not be generalized to other groups, e.g., high school students. Lastly, the study lasted

for eight weeks. Variables like attitude and computational thinking might require more time to be improved. Longer studies can be done to explore these variables.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 832-835.

Arnon, I., Cottrill, J., Dubinsky, E., Oktac, A., Fuentes, S. R., Trigueros, M., ... Weller, K. (2013). *APOS theory: A framework for research and curriculum development in mathematics education*. New York, NY: Springer.

Bell T. & Vahrenhold J. (2018). CS unplugged—how is it used, and does it work?. In: Böckenhauer HJ., Komm D., Unger W. (Eds), A*dventures between lower bounds and higher altitudes* (pp. 497-521). Springer, Cham.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini languages: a way to learn programming principles. *Education and information technologies*, *2*(1), 65-83.

Cakiroglu, U., Cevik, I., Koseli, E., & Aydin, M. (2021). Understanding students' abstractions in block-based programming environments: A performance-based evaluation. *Thinking Skills and Creativity*, *41*, 100888.

Cetin, I. (2020). Teaching loops concept through visualization construction. *Informatics in Education*, *19*(4), 589-609.

Cetin, I., & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *The Journal of Mathematical Behavior*, *47*, 70-80.

Chao, P. Y. (2016). Exploring students' computational practice, design, and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39.

Dijkstra E.W. (1982). S*elected writings on computing: a personal perspective*. Texts and Monographs in Computer Science. New York, NY: Springer.

Dubinsky, E. (1995). ISETL: A programming language for learning mathematics. *Communications on Pure and Applied Mathematics, 48*(9), 1027–1051

Green, S.B., & Salkind, N.J. (2013). U*sing SPSS for Windows and Macintosh: analyzing and understanding data*. New Jersey: Pearson.

Gul, D., Cetin, I., & Ozden, M. Y. (2022). A scale for measuring middle school students' attitudes toward programming. *Computer Applications in Engineering Education*, *30*(1), 251-258.

Herrmann, T. (2003). Learning and teaching in socio-technical environments. In *Informatics and the Digital Society* (pp. 59-71). Springer, Boston, MA.

Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research*, *58*(8), 1467-1493.

Huitt, W., & Hummel, J. (2003). Piaget's theory of cognitive development. *Educational psychology interactive*, *3*(2), 1-5.

Kandemir, C. M., Kalelioğlu, F., & Gülbahar, Y. (2021). Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices. *Computer Applications in Engineering Education*, *29*(1), 29-45.

Kert, S. B., Erkoc, M. F., & Yeni, S. (2020). The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Thinking Skills and Creativity*, *38*, 100714.

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly*, *92*(3), 170-181.

Korkmaz, O., Cakir, R., & Ozden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in human behavior*, *72*, 558-569.

Kölling, M. (2015). Lessons from the design of three educational programming environments: Blue, BlueJ, and Greenfoot. *International Journal of People-Oriented Programming (IJPOP)*, *4*(1), 5-32.

Kölling, M., Brown, N. C., & Altadmri, A. (2017). Frame-based editing. *Journal of Visual Languages and Sentient Systems*, *3*(1), 40-67.

Krendl, K. A., & Broihier, M. (1992). Student responses to computers: a longitudinal study. *Journal of Educational Computing Research, 8*(2), 215–227.

Major, L., Kyriacou, T., & Brereton, O. P. (2012). Systematic literature review: teaching novices programming using robots. *IET Software*, *6*(6), 502-513.

Mihci, C., & Ozdener Donmez, N. (2017). Teaching gui-programming concepts to prospective K12 ICT teachers: MIT App Inventor as an alternative to text-based languages. *International Journal of Research in Education and Science*, *3*(2), 543-559.

Mladenović, M., Mladenović, S., & Žanko, Ž. (2020). Impact of used programming language for K-12 students' understanding of the loop concept. *International Journal of Technology Enhanced Learning*, *12*(1), 79-98.

Moons, J., & Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education, 60*, 368–384.

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education*, *5*(2), 149-174.

Panskyi, T., Rowinska, Z., & Biedron, S. (2019). Out-of-school assistance in the teaching of visual creative programming in the game-based environment–Case study: Poland. *Thinking Skills and Creativity*, *34*, 100593.

Papert, S. (1980). *Mindstorms, Children, Computers, and Powerful Ideas*. New York, Basic Books.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.

Piaget, J. (1964). Cognitive development in children: Development and learning. *Journal of Research in Science Teaching 2*(3), 176-186.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67.

Román-González, M., Perez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678-691.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129–141.

Scherer, R., Siddiq, F., & Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, *109*, 106349.

Sullivan, A., & Bers, M. U. (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*(1), 3-20.

Tikva, C., & Tambouris, E. (2021). A systematic mapping study on teaching and learning Computational Thinking through programming in higher education. *Thinking Skills and Creativity*, *41*, 100849.

Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with scratch. *Computers & Education, 120*, 64–74.

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education, 18*(1), 1–25.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. M. (2011, February). Research notebook: Computational thinking- what and why? *The Link Magazine*, 20–23. Retrieved from https://www.scs.cmu.edu/link.

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. *Computer Science Education*, *29*(2-3), 177-204.

Zhang, Y., Luo, R., Zhu, Y., & Yin, Y. (2021). Educational robots improve k-12 students' computational thinking and STEM attitudes: systematic review. *Journal of Educational Computing Research*, 1-32.

Culturally Responsive Computing in Teacher Training: Designing Towards the Transformative Learning of Girls in STEM

Maha Elsinbawi[1]
Aaminah Norris[1]
Abigail Cohen[1]
Maureen A. Paley

[1]California State University Sacramento, CA, United States of America

DOI: https://doi.org/10.21585/ijcses.v6i2.179

Abstract

This paper reports on the findings of a Design-Based Research (DBR) study that investigated the transformative learning of six high school computer science teachers after they participated in a professional development (PD) training with a focus on Culturally Responsive Computing (CRC). Findings from the statistical analysis of pre-and post-surveys reveal ways in which teachers' understanding and enactment of CRC in their classrooms led to a reporting of increased student engagement, a deeper understanding of diverse learning needs, and improved access to cultural resources to specifically meet girls' needs. Findings from interviews and focus groups further reveal that after engaging in the PD, teachers qualitatively adapted their classroom strategies in order to uplift the cultural practices and gender identities of historically marginalized students. This study has implications for how teachers' professional development is designed and how they are guided to enact culturally responsive computing in ways that help recruit and retain racial and ethnic minority girls in CS courses.

Keywords: Culturally Responsive Computing, Teachers' Professional Development, Girls' participation in computer science, Intersectionality, Transformative learning

1. Introduction

The main goal of this research is to understand the impact of professional development in subject-specific Computer Science (CS) content using culturally responsive computing (CRC) on teachers' instruction of rigorous CS coursework and their ability to support students and underrepresented girls in the ICT pathways. The intervention described in this article is a concerted attempt at addressing a pipeline issue regarding the lack of diverse representation in CS, namely the enrollment of girls in this disciplinary domain ((NCES, 2020). From August 2019-September 2021, we conducted a Design Based research (DBR) that consisted of mixed methods study. In order to develop a High School and

Researcher-Practitioner Partnership (RPP), which is a crucial tool to sustain equity research (Vetter et al., 2022), faculty professors from the teaching credentials and computer science departments at a Northern California University, partnered with computer sciences teachers from a very diverse Northern California school district, which consists of various races and underrepresented minorities.

All computer science courses in this district are taught in six high schools' "Information and Communication Technology" (ICT) career pathways; career pathways are a rapidly growing reform movement in California, especially in high needs districts that enroll large numbers of low-income, diverse students. During the 2017-2018 academic year, forty-eight percent of the total high school population at these six schools were girls. During the 2017-2018 academic year, 155 girls participated in the six ICT pathways. This means that only 25% of students enrolled in ICT are girls. The rationale is to leverage girls and underrepresented students' shared interests in ICT, by infusing Culturally Responsive Computing in the content so that it becomes more engaging, appealing, and inclusive of them.

In the following article, we begin with a literature review outlining the empirical impetus for a project that privileges a Culturally Responsive approach to computer science learning and pedagogy. We continue with the findings from a design-based mixed-methods study that highlights how professional development that centers a critical approach to computer science and the development of cultural competence leads to the design of transformative learning experiences for girls. This article will conclude with implications for the training of computer science teachers and the development of learning ecologies that support young girls in STEM.

2. Literature Review

Culturally Responsive Computing (CRC) connects Ladson-Billings' (2014) theory of culturally relevant pedagogy (CRP) to the teaching of computing. CRP is a pedagogical approach designed to develop students' academic success, cultural competence, and sociopolitical consciousness by connecting curricular content to students' cultural understanding and real-world problem solving. Theorists suggest that using CRC in STEM learning environments can support student learning and address issues of power, race, and gender to help students (re)imagine their futures, especially for girls (Cheryan et al., 2015; Barton & Tan, 2010; Rosebery et al., 2016; Vakil, 2014). For example, Scott and White (2013) point out, "students' perception of their current cultural identities greatly influences the value they have for activities." These authors theorize that Culturally Responsive Computing (CRC) is a means to incorporate students' cultural identities into computer science teaching. Moreover, Scott et al., (2014) suggest that teachers should cultivate and establish their own cultural proficiency about students' identities and use this to build their lessons.

2.1 The Need for Culturally Relevant Pedagogies (CRP)

The National Center for Education Statistics (NCES) concluded that more than half of students of the global majority non-white races were enrolled in public schools in which less than a quarter of the students were of their own race. The NCES also reported the following regarding the minority composition of the public elementary and secondary classroom:

"In fall 2017, approximately 31 percent of public elementary and secondary students attended public schools in which the combined enrollment of minority students was at least 75 percent of total enrollment. More than half of Hispanic (60 percent), Black (58 percent), and Pacific Islander (53 percent) students attended such schools. In contrast, less than half of American Indian/Alaska Native students (39 percent), Asian students (39 percent), students of Two or more races (20 percent), and White students (6 percent) attended such schools." (NCES, 2020).

What these national statistics imply is that diversity within public schools is increasing, and the need for CRP application is becoming more crucial if schools intend to support all students' success.

The NCES also reported that in 2009, compared to boys, lower percentages of girls high school graduates reported that they liked mathematics or science (NCES, 2015). In the same year, 2009, NCES also emphasized the percentage of girls enrolled in computer/ information science was 13.8% compared to males whose percentage of enrollment was 24%. National Assessment of Educational Progress (NAEP) described the average mathematics and science scale scores of high school graduates who earned credits in STEM related technical courses, and specifically for computer/ information science in 2009, to be 155/300 for girls and 164/300 for boys (IES, 2009). These numbers and percentages indicate the lack of girls' interest and participation in computer/ information science at the high school level. Although the reasons why this gender discrepancy exists are beyond the scope of this paper, the tools of remedy and the means to create equity still need to be investigated. Often, the trend in CS/CRP research studies was to focus on the importance of enhancing girls' participation and increasing diversity in the computer science class, but the techniques on how to achieve this were not researched in depth. Our paper intends to fill that gap by emphasizing Culturally Responsive Computing (CRC) as a tool that helps construct justice between genders in computing, and increases the interest and participation of girls, as a minority group, in computer science. To understand what CRC is, it is crucial to dig deeper into one of the main foundations of this concept: Culturally Responsive Pedagogies.

Culturally Responsive Pedagogy (CRP), also sometimes called Culturally Relevant Teaching as emphasized by Ladson-Billings (1995), is a concept that originally started to research teachers who had excelled African American students. Ladson-Billings (1995) has emphasized three criteria that need to exist in the students in order to apply CRP: Students must experience academic success; students must develop and/or maintain cultural competence; and students must develop a critical consciousness through which they challenge the status quo of the current social order. The first principle of this

definition depends on the students to prove their academic accomplishment, even though they may face hostility both in and out of the classroom. In the second pillar described by Ladson-Billings (1995), the students need to show and be proud of their own culture, and the teachers should try to learn about this culture through the students; for example, some teachers allow the students to choose their own music and use their home language in the classroom. The third component of applying CRP is the critical consciousness that allows the students to be aware of what is suppressing their freedom within society and be able to criticize it and fight it back. Another definition of CRP was noted by Brown-Jeffy and Cooper (2011). They emphasized the principles of CRP as comprised of 5 main components: (1) Identity and achievement, which takes the unique culture and identity of the students into setting the curriculum, (2) equity and excellence that ensures there is equal access for all, (3) developmental appropriateness where psychological needs, motivation, collaboration, and engagement are met, (4) teaching the whole child which is equivalent to empowering the students, and finally (5) student-teacher relationship that needs to be caring and interacting (Brown-Jeffy & Cooper, 2011).

The positive effects of Culturally Responsive Pedagogy (CRP) were discussed in several research papers. Milner (2011), for example, summarized the main outcome of CRP as empowering the students. He explicates the details of the impact of CRP as follows: to empower students by allowing them to participate in the deconstruction and construction of the curriculum given to them, which in turn highlights any inequities and ultimately leads to students' academic successes. CRP also allows for the incorporation of students' culture. An incorporation which transcends the negative effects of the dominant culture and eventually creates classroom contexts that are innovative and focused on meaningful student learning (and consequently academic achievement) by strengthening the cultural competence (Milner, 2011). Other research highlights CRP's positive effects when taught at the pre-service level for teachers, stating that when teachers are encouraged to reflect on their own racial and cultural identities, there is an improvement in the connections made with diverse groups of students. (Howard, 2003; Siwatu, 2007).

2.2 Culturally Responsive Computing (CRC)

Culturally Responsive Computing emerged as a potential approach for successfully engaging marginalized and underrepresented students in technology (Scott et al, 2015). Drawing from the definition and components of Culturally Responsive (or Relevant) Pedagogy, CRC shares the same three pillars (based on Ladson-Billings' work): asset building, reflection, and connectedness. CRC builds on these pillars with a particular focus on technology education. Scott et al, (2015), defined specialized points of interest for the CRC to focus on as follows:

"(1) Motivate and improve science, technology, engineering, and math (STEM) learning experiences; (2) Provide a deeper understanding of heritage and vernacular culture, empowerment for social critique,

and appreciation for cultural diversity; (3) Bring points A and B together: to diminish the separation between the worlds of culture and STEM; (4) This technology must not only respond to these identity issues, but also satisfy pedagogical demands of the curriculum."

These points need to be directed towards marginalized groups of students in order to include all the underrepresented parties in the education of technology (Scott et al, 2015). In other words, the unique cultural background of the marginalized students' needs to be understood and taken into consideration in regard to the development of the curriculum in general and the STEM classes in particular.

Culturally Responsive Computing is then a concept that is trying to include all identities (gender, culture, ethnic, religious, etc.) into consideration to improve the Computational Thinking skills of the students. The positive effects of CRC were also emphasized by several studies. Research suggests that using CRC in STEM learning environments can support student learning and address issues of power, race, and gender in order to help students in general and marginalized students specifically (re)imagine their futures (Ryoo, 2019; Barton & Tan, 2010; Ford, 2014; Rosebery et al, 2016; Morales-Chicas et al, 2019). Other research emphasized the idea that CRC supports the connection between school and community in ways that incorporate the knowledge and skills of underrepresented communities into math and computing education, while paving the direction to allow technologies to encourage education-based social movement (Lachney, 2016; Eglash et al, 2013).

Ashcraft et al, (2017) highlighted through their research on COMPUGIRLS, how the implementation of CRP positively affected girls in computing. They allowed the girls to transform from silent receivers to active contributors in their own educational process. Roque et al. (2021), also concluded the positive effects of CRP on historically marginalized students by including both the students and their families in creative learning programs for computational construction kits by using new possibilities of their storytelling, Litts et al, (2021), also emphasized the effectiveness of using "storytelling" as a culturally responsive tool for computing. This research also emphasizes the importance of CRP, but by focusing on educating the teachers with the concepts of CRC, in order to implement CRP in their computer science classrooms. Pozos et. al. (2022) highlighted the importance of introducing justice-oriented curricula to multilingual students to be more responsive to the computational thinking materials, by using a case study approach, they were able to reach 3 principles to be used by teachers.

2.3 Importance of Intersectionality in Computer Science Class

Pournaghshband and Medel (2020) emphasized the concept of intersectionality while studying the phenomenon of girls' underrepresentation in computer science, a concept that was disregarded by other research. In the rise of diverse classrooms, girls may now be underrepresented not only because of their gender but also because of their cultural backgrounds. The merge of two or more-dimensional identities is what we refer to as intersectional identity. The authors concluded that Culturally Responsive

Computing is a concept that groups all identities and hence, intersectionality is included in CRC. Intersectionality is defined as the interaction between several social identities like for example, race, class, and gender in cultivating life experiences, especially those experiences of oppression (Gopaldas, 2013; Mehrotra, 2010).

Throughout primary and secondary education, girls are underrepresented in most fields of computer science. But little research tackled how CRC can be a tool to help close the gap of underrepresentation of girls in general (especially high school girls students). For example, Searle and Kafai (2015) emphasized the positive effect CRC may have on girls from indigenous communities. They endorsed the fact that making sense to the students is a key to their success in academics in general and underrepresented girls in specific, and they also encouraged the idea of applying CRC not only in computing but also in educational crafts making activities (as a software) that can help address the "identity gap" for girls and students from non-dominant backgrounds (Searle & Kafai, 2015). A point also tackled by Corkin et al, (2020), who examined the extent to which an intervention informed by culturally relevant pedagogy theory predicted the motivation of underrepresented high school students to take computer science courses.

Culturally Responsive Computing is considered to be a means that can be used by students to understand their own intersectionality and by teachers in order to better understand their students' unique identities. The general aim of this research is to emphasize the positive effects of teaching computer science teachers the principles of CRC for the implementation of CRP and the improvement of the classroom culture, and the inclusion of marginalized minorities, especially girls. We used a systematized way to implement CRC by infusing transformative learning for the teachers through the use of a DRB. The implementation will be emphasized in the method section.

3. Method

This paper is a result of the collaboration between researchers and practitioners in the context of a Design-Based Research framework. Researchers have argued that the RPP model is effective because "collaboration is one of the best ways to close the research/practice gap and propel more evidence-based practice (Murray, 2017)" and also because the communication between researchers and practitioners can be the most beneficial for the two communities, although the researchers part still carry the lion's share (Sato & Loewen, 2022). We conducted a mixed methods study of six CS teachers (all from the same school district) over a two-year period. Data sources consisted of in-depth interviews and focus groups. Pre- and post-surveys were also administered, where respondents answered both demographic questions and self-identified their culturally responsive pedagogical practice The in-depth interviews were conducted at the beginning of the first year, and approximately at the same time, the teachers were asked to fill out the online survey. The research team introduced various concepts to the teachers over the course of the two years through quarterly sessions of Professional developments and trainings that

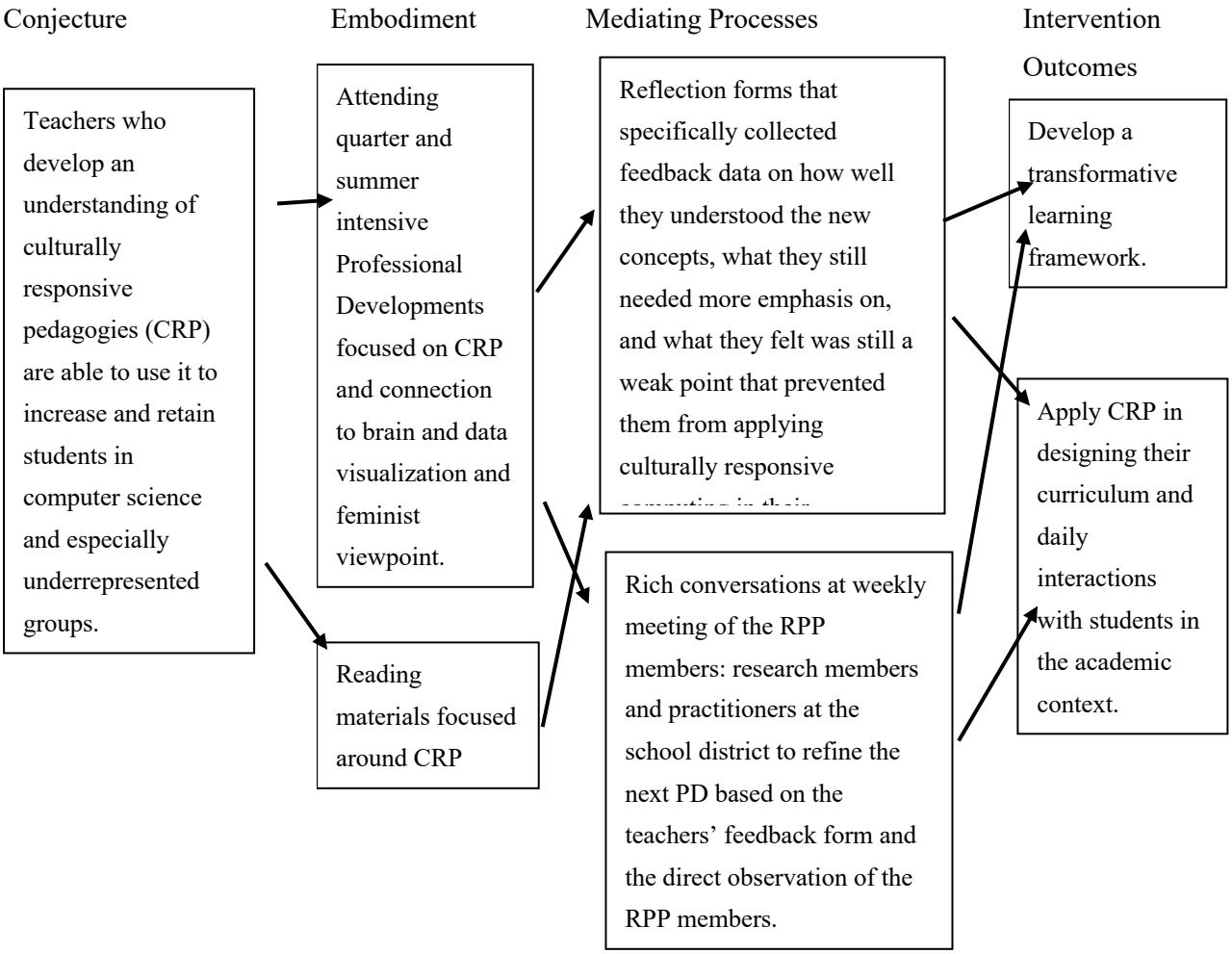
included culturally responsive computing, the relationship with the brain, data feminism visualization, simulations and hackathons. At the end of the second and last year, focus groups were conducted and the post-survey was administered to the teachers.

3.1 Design Based Research

As a Social Design Based Experiment (Gutierrez & Jurow, 2016) we engaged in concerted side-by-side stakeholder participation in all aspects of the project. Importantly, we were keenly aware of how teachers influence the learning process of the students the most, putting theory into practice to alter their methods in the classroom. Since we are aware that a one size fits all mentality, will not serve the students especially when learning computer science and STEM in general, a design-based research allowed the teachers to reflect on new information and material given to them through mutual relations of exchange; and in this case the culturally responsive computing that takes into consideration and prioritizes the unique characteristics of the students (Gutierrez & Vossoughi, 2010). In order to create the desired outcomes, the research team became a central part of the teachers' learning ecology, by first being points of contact for any inquiry throughout the research and second by designing professional development opportunities that helped to develop the new enactment for transformative learning.

In our work we considered how design is a re-mediating activity (see Gutiérrez, 2018) that consists of making a shift in the way the entire ecology for learning (contexts, tools, relationships, etc.) must be engaged in order to address learning at a systemic level. Thus, we went beyond emphasizing the development of technical disciplinary skill(s) for our teacher participants, but we instead tried to shift their way of thinking on how they viewed their students in general and the students who identified as girls in particular. We were keen on collecting the teachers' thoughts and feedback after each Professional development (see Figure 1 below).

Figure 1: Conjecture
Map*



*Based on the work of Sandoval (2014)

3.2 Participants

Six computer science teachers (N=6; three males and three females) participated in the study. All Six participants were given numbers and pseudonyms, i.e., teacher 1: Robert, teacher 2: Catherine, etc. (see table 5 for participants' demographics). Throughout the two-year period, the teachers received CRC training, with a focus on equity in computer science (CS) for girls to support teachers' instruction of rigorous CS content to their girl and underrepresented students. The ages of the participants at the time of the research ranged from 36 to 62 years old. All the teachers had five or more years' of experience. Four teachers taught in Title I schools (see Table 1).

Table 1. Teachers' demographic at the time of the research

Teacher number	Pseudonym	Gender	Age	Years of Experience
Teacher 1	Robert	Male	58	5 years and more
Teacher 2	Catherine	Female	62	5 years and more
Teacher 3	Caleb	Male	47	5 years and more
Teacher 4	Luna	Female	50+	5 years and more
Teacher 5	George	Male	46	5 years and more
Teacher 6	Julia	Female	36	5 years and more

4. Data Collection and Procedures

This mixed methods study of six computer science teachers conducted over two years examines the effects of Culturally Responsive Computing (CRC) on student engagement and teachers' knowledge of their girls and underrepresented students' needs focusing on their identity and intersectionality. Qualitative data sources include in-depth interviews and focus groups. Quantitative data was collected via a survey.

4.1 The Qualitative Part

4.1.1 The Interviews

Participants were interviewed virtually through Zoom for an average of two hours per interview. Based on previous interview methods recommendations ((Velardo and Elliott, 2021), each semi-structured interview had one interviewer from the research team, in addition to a silent observer, also from the research team, who was taking field notes and attending the interview silently, i.e., with their camera and audio off. The focus of the in-depth interviews was to explore ways that the teachers handle students' unique identities and intersectionality (including gender, culture, language, and race) and how they affirm this uniqueness. There was also a focus on learning if and how teachers help students to develop pride, confidence, and healthy self-esteem without denying the value and dignity of others, and their perceptions and techniques that strengthen diversity in the classroom. Questions included teachers' recruitment strategies, especially with girls in their ICT programs and how they apply equity and social justice in their classroom, specifically in regard to their (intersectional) gender and social identities. The opinions on PDs and how they affected their teaching strategies were also discussed.

4.1.2 The Focus Group

Two focus groups were conducted virtually via Zoom and had an average of 3-4 participants. Similar to the interviews, the focus groups were semi-structured, and lasted for almost two hours. The focus groups mainly emphasized teachers' strategies in applying equity in their classrooms with regard to the differences in culture, and how they addressed the issues of gender in Computer Science.

4.1.3 The Transformative Learning Part

The transformative learning consisted of 4 quarterly Professional Developments (PD), and one summer intensive session over the course of two years. The PDs were two hours long each, and the summer intensives were 4 hours each for a period of 5 days. The PDs were developed by the research team: The Principal investigator, who is an expert in culturally responsive pedagogies, the co-PI who is an expert in computer science, two research assistants and one research coordinator. The team at the school's district was also involved in every step of the transformative learning process. The PDs focused on explaining the concepts of Culturally Responsive Computing, the relationship with the brain and data visualization. The first summer intensive focused on an introductory background in data science including the analytical pipeline of data collection processing, analysis, and data visualization. The second summer intensive focused on relating data visualization through a data feminist viewpoint (authors, 2021). The main goal was to relate the concepts of computing to culturally responsive pedagogies that specifically frame girls and their unique ways of understanding the data, through re-mediating activities, by emphasizing contradictions, history, and equity. Each PD gave the teachers' participants the chance to reflect and re-mediate by filling in a feedback form. This form highlighted the new concepts they were able to grasp, what they still needed more emphasis on, and what they felt was still a weak point that prevented them from applying culturally responsive computing in their classrooms.

As detailed by authors (2021), the design of the professional development week is to teach the participants data science by applying 3 data feminist principles, by incorporating them into the lessons plan. with the aims to alter the traditional approach by following the three tenants summarized as follows: "1) invent new ways to represent data unknowns, 2) invent new ways to reference the material economy behind the data, 3) make dissent possible." The five days planned by the research team included activities ranging from data science, community building, and discussions.

4.2 The Quantitative Part: The Survey

The participants completed two online surveys: the pre-test, at the beginning of the first year, then the post-test at the end of the second year. The pre-surveys were administered before the transformative

learning part took place; we wanted to quantitatively test if there was any improvement to the teachers' knowledge before and after they were exposed to the culturally responsive computing PDs.

4.2.1 The Online Survey

The survey was adapted from a CRP rubric created by the Centennial School District in Oregon. This district has one of the highest numbers of homeless students at a percentage of 1.6%. The original survey: Centennial School District Culturally responsive rubric, which had 4 main sections: planning and preparation, the classroom environment, instruction and professional responsibilities. We recreated the survey with a focus on measuring the teachers' culturally responsive pedagogical knowledge, and the degree of its application in the classroom with their students. Our survey aimed to measure specific factors, as illustrated in table 2.

Table 2. Factors components of the teachers' survey

Factor 1: Knowledge of child and adolescent development	Factor 2: Knowledge of the learning process
Factor 3: Knowledge of students' skills, knowledge, and language proficiency	Factor 4: Knowledge of students' interests and cultural heritage
Factor 5: Knowledge of students' diverse needs	Factor 6: Knowledge of content related pedagogy
Factor 7: Appropriateness for diverse learners	Factor 8: Resources for classroom use
Factor 9: Resources to extend content knowledge and pedagogy	Factor 10: Resources for students
Factor 11: Teacher interactions with students	Factor 12: Student interaction with other students
Factor 13: Expectations for learning and achievement	Factor 14: Teacher creates environment that promotes pride in work
Factor 15: Student engagement	

4.2.2 Cronbach's Alpha Reliability Test

A Cronbach's Alpha test was run for each Likert Scale item within the pre and post teachers' survey to determine the internal consistency and whether it could produce reliable composite scores. The higher α coefficient > 0.6, the more the items have shared covariance and probably measure the same underlying concept. Results showed Alpha α in all of the category variables is > 0.6, which means the test has high internal consistency and acceptable index (Nunnally and Brenstein, 1994). The Pre survey yielded an alpha value of ά =0.824 >0.6. (Table 3).

Table 3. Test Cronbach's alpha, Reliability Statistics

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.824	.803	15

5. Data Analysis

Data analysis of the impact of teacher transformative learning on the implementation of CRC was performed. Qualitative data concerning the interviews was analyzed first, due to the fact that this was the first data collection method used. We then analyzed the two focus groups, and lastly conducted the quantitative analysis using the paired sample T-test after having the data from the pre and post-surveys. We relied on Corbin and Strauss' (2008) constant comparative method of grounded theory. Open and axial coding were used to classify concepts and codes under various categories to extract emergent themes (Creswell, 2014). Some of the main themes and codes we discovered in the interviews with our 6 teachers participants were: Girls' recruitment strategies, girls learning styles, teachers' styles to support students, equity and justice in computer science classes, gender and sexism in computer science classes, cultural and race identities, types of support to help the teachers. From the focus groups, other themes were revealed like for example: gender, sexism and girls in computer science, Difficulties with Discussion of Gender/Sexism in computer science, challenges girls face in computer science classes, challenges minority groups to face in computer science classes, Curriculum Changes to Build Culture of CS where Girls Feel a Part of, Extracurricular Changes to Build Culture of CS where Girls Feel a Part of, the district's role to build Culture of CS where Girls Feel a Part of.

Preliminary findings and generalizations were extracted from the analysis and then compared to the existing literature. Transcripts were edited for mistakes and consistency of definitions of codes. Researchers' reflexivity, position, and biases were discussed for reliability and peer debriefing and member-checking were used for trustworthiness. (Creswell, 2014; Denzin & Lincoln, 2011). The

transformative part took place after the pre-survey was administered, and right after the first batch of individual interviews. After each Professional development, a feedback form was administered to the teacher and included a part on what was most/ least useful, and are you ready to use CRC in your classroom, and if not what is still missing. We used the information from the feedback forms to develop the next PD session in a way that tackled the missing and lacking points emphasized by the teachers. The pre- and post-survey results were first transferred from Qualtrics to SPSS, and from there, a Paired sample T-test was analyzed to measure any increase in the Means of the factors components of the survey (Table 4). Because we were using the same test on the same sample, the paired sample T-test seemed the most suitable means of statistical analysis.

Table 4. Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Pre-Knowledge of child and adolescent development	2.67	6	.516	.211
	Post-Knowledge of child and adolescent development	3.00	6	.000	.000
Pair 4	Pre-Knowledge of students' interests and cultural heritage Click to write the question text	2.67	6	.516	.211
	Post-Knowledge of students' interests and cultural heritage Click to write the question text	3.00	6	.000	.000
Pair 10	Pre-Resources for students	2.50	6	1.225	.500
	Post-Resources for students	3.00	6	1.095	.447
Pair 15	Pre-Student engagement	2.83	6	1.169	.477
	Post-Student engagement	3.33	6	.816	.333

6. Results and Discussion

We triangulated the quantitative and qualitative data to develop a deeper and richer understanding of the impact of CRC on teachers. A paired sample t-test was conducted to compare teachers' knowledge of CRC before and after the training. The results suggested that the mean of Four Factors increased, which indicates an improvement resulting from the CRC training provided to the teachers, and their knowledge increased, which also confirms the results by Leonard and Sentance (2021).

6.1 Knowledge of Child and Adolescent Development

We found an increase in the mean for the Knowledge of Child and Adolescent Development factor, IV (Independent Variable) level 1 (M= 2.67, SD= 0.516), and IV level 2 (M= 3.00, SD= 0.00). This indicates that teachers' knowledge of the developmental characteristics of their students improved including the impact of students' race, gender, and culture on their development. The increase in this factor highlights the teachers' deeper understanding of how the identities and cultures of the students affect their growth. This is an integral component of CRC. Before the CRC training, teachers did not express the need to incorporate their knowledge of their diverse students into their teaching. For example, Robert mentions:

"Yeah, my class is very diverse. I have one Caucasian girl in my class and everybody else of every other race on the planet. And I don't have to work very hard at the cultural part. Sometimes I have to work hard about maybe getting them to work in different groups, but they kind of get a little, they have little packs of three, four, five kids and they kind of just work together and I don't, I don't try to monitor. I just see what's going on in the classroom. I don't have to work very hard at the cultural part."

As Robert's comment reveals, prior to the professional development, the teachers participating in this study were aware of the diversity in their classrooms. However, the in-depth interviews and focus groups showed that they did not explicitly address the various needs of their diverse learners before the professional development, nor did they acknowledge the deeper need to focus on their students' race and culture. The reasons they mentioned were that they did not think it was important or because they did not have enough background to start this conversation with the students.

6.2 Knowledge of Students' Interests and Cultural Heritage

We noticed an increase in the mean of the Knowledge of Students' Interests and Cultural Heritage factor, IV level 1 (M= 2.67, SD= 0.516), and IV level 2 (M= 3.00, SD= 0.00). Teachers had an increased recognition of the importance of understanding their students' interests and cultural heritage. Additionally, teachers also revealed that they understood the importance of knowing the individual needs of their diverse students in general, and their girl students. A problem that exists, as discussed by Scott et al., (2014) is the obvious gap in the CRC theory that leaves out cultural identity, and instead

focuses on technical literacies. This practice sidelines minoritized groups from being engaged by the pedagogies. Our teachers, too, experienced discomfort when talking about culture. According to Robert, “That's always been hard to try to wiggle those in there." Caleb, also, had a similar response:

“I don't know that I necessarily [discuss diversity] explicitly. I guess I should. But I don't actually get into that topic. I pretty much stay with ‘this is the code we're going to use,’ and I demonstrate it. […] I don't spend a lot of time in my classes with, you know, expressing that topic other than to tell them what my experience was when I went to college.”

We found that after the teachers received the CRC training, teachers are more aware of the different interests of the students and their diverse cultural heritage, resulting in serious efforts to encourage girls and accommodate their special interests. Catherine points out her experience by comparing her awareness levels before and after the PD by explaining:

“Before the PD I think I was less aware of the situation. I’ve always known that there was a […] lack of participation by girls, but I think that being part of what I call the program, has caused me to think more and be more aware every day when I teach. So, I try to relate what I learn from this [PD] and try to apply as much as I can, where I feel comfortable. […] before I feel like I couldn’t. " The teachers’ remarkable increasing awareness of the diverse needs, cultures and interests of their students is reflected in their interaction with them.

6.3 Resources for Students

The mean of the Resources for Students factor also increased, IV level 1 (M= 2.5, SD= 1.225) and IV level 2 (M= 3.00, SD= 1.095). Teachers showed improved knowledge of resources that appropriately reflect the gender identity and gender diversity of their students, including those available through the school or district, in the community, and on the internet. After the training, teachers were convinced of the need for different resources to accommodate students from various cultures. They were consequently ready and willing to search for additional resources to engage their diverse students. Robert, for example, who, at the start of this research did not feel comfortable discussing students’ identities, added cultural resources by inviting different speakers to his classroom.:

“So, I'm trying to embrace [my students’] differences, and you know, the biggest way is not made in the regular classroom setting, but I guess it's more about people that my speakers talk about. These are the qualities they may have that people don't realize that aren't being, they're not being pronounced in the class of being coding, but in the workplace, those differences which could be very valuable.” The teachers embrace their students’ differences by searching for resources to accommodate their various cultural differences.

6.4 Student Engagement

The mean of the Student Engagement factor increased; IV level 1 (M= 2.83, SD= 1.169) and IV level 2 (M= 3.33, SD= 0.816). Teachers effectively employed strategies to ensure that all voices are heard in their classrooms. Teachers' focus on engaging all the students in the classroom increased after the CRC training. Specifically, teachers focused on engaging girls:

"[The PD] helped to force me out of my shell, to talk more to the students and try to get their personalities to shine through. And acknowledge those who are sharing both in the chat and by speaking. And I think, in a virtual environment it helps all students, both male and female to feel comfortable to share because they can share either as a group where they can share independently or privately to the teacher. So, I think that's really helped. I have noticed from last year to this year, an increase in female enrollment and I'm hoping that that trend just continues."

Overall, we discovered ways CRC raised teachers' awareness of diversity. Julia explains how the PD made her look for more diversity in the classroom, and search to add more diverse students and engage them because of how she came to believe this enriches the teaching experience. She exemplifies this by saying:

"It definitely made me think a lot about what I'm doing. […] So, I've always noticed that, you know, I like to get a lot more diversity in my classroom. And it definitely made me think about it a lot more. And I'm trying to, I'm still, feel like I have a long way to go, but. They definitely opened my eyes a little more."

Therefore, we uncovered that incorporating CRC practices into CS instruction engages diverse students in general and girls. CRC supports the interactions between teacher/ student and student/ student because it results in understanding students 'differences and shaping the CS content to address their diverse needs.

7. Scientific Significance of the Study

This study reveals that training in Culturally Responsive Computing (CRC) positively affects teachers' ability to engage girls in computing (see figure 1 for the conjecture map). Our results also align with Scott et al., (2014), who emphasize the obvious gap in the CRC theory that leaves out cultural identity, and instead focuses on technical literacies. Teachers' knowledge of child and adolescent development, knowledge of students' interests and cultural heritage, their willingness to expand the various resources that respect the students' differences, and increased students' engagement highlight the importance of training teachers in CRC. We see this research as confirming the importance of CRC or the use of culturally responsive pedagogies in computer science, which comes in accordance with Brown et al, (2019), who also endorsed in their research the importance and positive effects of teaching CRP to

STEM teachers. Transformative learning in the form of social design research allowed the teachers to better understand and acknowledge the students' intersectionality by learning about their diverse cultures, ethnicities, and backgrounds and how it affects their social and academic development. This creates a positive and inviting classroom culture that results in more equitable opportunities for the students' learning, especially marginalized and underrepresented ones like girls in the STEM area.

Acknowledgment

The authors wish to thank Dr. José Lizárraga for their support and assistance in the writing of this manuscript. We also wish to thank our research-practice partnership members for their willingness to engage in equity-centered design conversations.

This material is based upon work supported by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Ashcraft, C., Eger, E., & Scott, k. (2017). Becoming technosocial change agents: Intersectionality and culturally responsive pedagogies as vital resources for increasing girls' participation in computing. Anthropology & Education Quarterly, 48(3), 233–251. http://doi.org/10.1111/aeq.12197

Barton, A. C., & Tan, E. (2010). We be burnin'! Agency, Identity, and Science Learning. The Journal of the Learning Sciences 19(2), 187-229.

Baynes, A & Norris, A. (2021). Professional development for high school computer science teachers on data science through a Feminist Lens. IEEE Frontiers in Education Conference (FIE), 1-5, http://doi.org/10.1109/FIE49875.2021.9637249

Brown, A. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. The Journal of the Learning Science, 2(2), 141-178.

Brown, B., Boda, P., Lemmi, C., & Monroe, X. (2019). Moving culturally relevant pedagogy from theory to practice: Exploring teachers' application of culturally relevant education in science and mathematics. Urban Education, 54(6), 775-803. http://doi.org/10.1177/0042085918794802

Brown-Jeffy, S., & Cooper, J. (2011). Toward a conceptual framework of culturally relevant pedagogy: An overview of the conceptual and theoretical literature. Teacher Education Quarterly, 65-84. Retrieved from: https://files.eric.ed.gov/fulltext/EJ914924.pdf

Centennial School District 28J. Revised Culturally Responsive Teacher Rubric. Fully Revised Culturally Responsive Danielson Rubric (csd28j.org)

Cheryan, S., Master, A., & Meltzoff, A. N. (2015) Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. Frontiers in psychology, 6(49). https://doi.org/10.3389/fpsyg.2015.00049

Corbin, J., & Straus, A. (2008). Basics of qualitative research (3rd ed.). Sage.

Creswell, J. W. (2014). Research design: Qualitative, Quantitative, and mixed methods approaches (4th ed.). Sage.

Denzin, N. K., & Lincoln, Y. S. (2011). The sage handbook of qualitative research. Sage.

Eglash, R., Gilbert, J., Taylor, V., & Geier, S. (2013). Culturally responsive computing in urban, after-school contexts: Two approaches. Urban Education, XX(X), 1-28. http://doi.org/10.1177/0042085913499211

Gopaldas, A. (2013). Intersectionality 101. Journal of Public Policy & Marketing, 32 (Special Issue), 90–94

Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: the exploring computer science program. ACM Inroads, 3(2), 47-53.

Gutiérrez, K. D. (2008). Developing a sociocritical literacy in the third space. Reading research quarterly, 43(2), 148-164.

Gutiérrez, K & Vossoughi, S. (2010). Lifting off the ground to return anew: Mediated praxis, transformative learning, and social design experiments. Journal of Teacher Education 61(1-2) 100–117. http://doi.org/10.1177/0022487109347877

Gutiérrez, K. D., & Jurow, A. S. (2016). Social design experiments: Toward equity by design. Journal of the Learning Sciences, 25(4), 565-598.

Gutiérrez, K. D. (2018). Social design–based experiments: A proleptic approach to literacy. Literacy Research: Theory, Method, and Practice, 67(1), 86-108.

Howard, T. (2003). Culturally relevant pedagogy: Ingredients for critical teacher reflection. Theory into Practice, 42(3), 195-202.

Kimberly A. Scott, Kimberly M. Sheridan & Kevin Clark (2014) Culturally computing: A theory revisited, Learning, Media and Technology, 40(4), 412-436, http://doi.org/10.1080/17439884.2014.924966

Ladson-Billings, G. (2014). Culturally Relevant Pedagogy 2.0: a.k.a. the Remix. Harvard Educational Review, 84(1), 74–84. http://doi.org/10.17763/haer.84.1.p2rj131485484751

Ladson-Billings, G. (1995). But that's just good teaching! The case for culturally relevant pedagogy. Theory into Practice, 34(3). 159-165. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.4239&rep=rep1&type=pdf

Lachney, M. (2016). Culturally responsive computing as brokerage: Toward asset building with education-based social movements. Learning Media and Technology, 42(4), 420-439, http://doi.org/10.1080/17439884.2016.1211679

Leonard, H. C., & Sentance, S. (2021). Culturally-relevant and responsive pedagogy in computing: A Quick Scoping Review. International Journal of Computer Science Education in Schools, 5(2), 3–13. http://doi.org/10.21585/ijcses.v5i2.13

Litts, B., Searle, K., Brayboy, B., & Kafai, Y. (2021) Computing for all: Examining critical biases in computational tools for learning. British Journal of Educational Training. 52(2). 842-857. http://doi.org/10.1111/bjet.13059

Mehrotra, G. (2013). Toward a continuum of intersectionality theorizing for feminist social work scholarship. Journal of Women and Social Work, 25(4), 417-430. http://doi.org/10.1177/0886109910384190

Milner IV, H. (2011). Culturally relevant pedagogy in a diverse urban classroom. Urban Re, 43, 66-89. http://doi.org/10.1007/s11256-009-0143-0

Morales-Chicas, J., Castillo, M., Bernal, I., Ramos, P., & Guzman, B. (2019). Computing with relevance and purpose: A review of culturally relevant education in computing. International Journal of Multicultural Education, 21(1), 125. http://doi.org/10.18251/ijme.v21i1.1745

Murray Law, B. (2017). Pillars of a Researcher-Practitioner Partnership. ASHA Leader, 22(8), 64. Retrieved from http://proxy.lib.csus.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=rzh&AN=124286161

National Center for Education Statistics (2020), Racial/Ethnic Enrollment in Public Schools, chapter one, retrieved from: http://nces.ed.gov/programs/coe/pdf/coe_cge.pdf

National Center for Education Statistics (2015), Gender differences in science, technology, engineering, and mathematics (STEM) interest, credits earned, and NAEP performance in the 12th Grade. Stats in Brief, 075. Retrieved from: http://nces.ed.gov/pubs2015/2015075.pdf

Nunnally, J.C., & Bernstein, I.H. (1994). The assessment of reliability. Psychometric Theory, 3, 248-292.

Pournaghshband, V., & Medel, P. (2020). Promoting diversity-inclusive computer science pedagogies: A multidimensional perspective. Association for Computing Machinery, Innovation and Technology in Computer Science Education (ITiCSE) ‘20, June 15–19, 2020, Trondheim, Norway. http://doi.org/10.1145/3341525.3387360

Pozos, R., Severance, S., Denner, J., & Tellez, K. (2022). Exploring design principles in computational thinking instruction for multilingual learners. Teachers College Record 124(5). DOI: http://doi.org/10.1177/01614681221104043

Roque, R., Tamashiro, M.A., Mcconnell, K., & Granados, J. (2021). Opportunities and limitations of construction kits in culturally responsive computing contexts: Lessons from Scratch Jr. and family creative learning. Interaction Design and Children (IDC '21), June 24–30, 2021, Athens, Greece. ACM, New York, NY, USA. http://doi.org/10.1145/3459990.3460728

Rosebery, A. S., Warren, B., & Tucker-Raymond, E. (2016). Developing interpretive power in science teaching. Journal of Research in Science Teaching, (10), 1571. http://doi.org/10.1002/tea.21267

Ryoo, J. J. (2019). Pedagogy that Supports Computer Science for All. ACM Transactions on Computing Education, 19(4), 1–23. http://doi.org/10.1145/3322210

Sandoval, W. (2014). Conjecture mapping: An approach to systematic educational design research. Journal Of The Learning Sciences, 23(1), 18-36.

Sato, M., & Loewen, S. (2022). The research–practice dialogue in second language learning and teaching: Past, present, and future. The Modern Language Journal, 106(3), http://doi.org/10.1111/modl.12791 0026-7902/22/509–527

Scott, K. A., & White, M. A. (2013). COMPUGIRLS' standpoint: Culturally responsive computing and its effect on girls of color. Urban Education, 48(5), 657–681. Retrieved from: http://journals.sagepub.com/doi/pdf/10.1177/0042085913491219

Scott, K., Aist, G., & Zhang, X. (2014). Designing a culturally responsive computing curriculum for girls. International Journal of Gender, Science and Technology, 6(2), 264- 276. Retrieved from: http://genderandset.open.ac.uk/index.php/genderandset/article/view/361

Scott, K., Sheridan, K., & Clark, K. (2015). Culturally responsive computing: A theory revisited. Learning Media and Technology, 40(4), 412-436. http://doi.org/10.1080/17439884.2014.924966

Searle, K., & Kafai, Y. (2015). Culturally responsive making with American Indian girls: Bridging the identity gap in crafting and computing with electronic textiles. GenderIT '15: Proceedings of the Third Conference on GenderIT, 9–16. http://doi.org/10.1145/2807565.2807707

Siwatu, K. (2007). Preservice teachers' culturally responsive teaching self-efficacy and outcome expectancy belief. Teaching and Teacher Education, 23, 1086–1101. http://doi.org/10.1016/j.tate.2006.07.011

Vakil, S. (2014). A critical pedagogy approach for engaging urban youth in mobile app development in an after-school program. Equity & Excellence in Education, 47(1), 31-45.

Velardo, S., & Elliott, S. (2021). Co-Interviewing in qualitative social research: Prospects, merits and considerations. International Journal of Qualitative Methods, 20, http://doi.org/10.1177/16094069211054920

Vetter, A., Faircloth, B. S., Hewitt, K. K., Gonzalez, L. M., He, Y., & Rock, M. L. (2022). Equity and social justice in research practice partnerships in the United States. Review of Educational Research. DOI: http://doi.org/10.3102/00346543211070048

What Emotions Do Pre-university Students Feel when Engaged in Computational Thinking Activities?

Rafael Herrero-Álvarez[1] , Coromoto León[1] , Gara Miranda[1] , Eduardo Segredo[1] , Óscar Socas[1] , María Cuellar-Moreno[2] , Daniel Caballero-Juliá[3]

[1] Departamento de Ingeniería Informática y de Sistemas of Universidad de La Laguna, SPAIN

[2] Departamento de Didácticas Específicas of Universidad de La Laguna, SPAIN

[3] Departamento de Didáctica de la Expresión Musical, Plástica y Corporal of Universidad de Salamanca, SPAIN

DOI: https://doi.org/10.21585/ijcses.v6i2.180

Abstract

Emotions play a crucial role in knowledge acquisition and can significantly impact motivation when studying a new field. Unfortunately, young people, especially girls, are often not drawn to Computer Science. To address this issue, we conducted an analysis of emotions among 8-9-year-old and 12-13-year-old students engaged in Computational Thinking activities, considering educational level, gender, and type of intervention. Our study sought to understand the lack of interest by examining the emotions present in primary and secondary school students. Hour-long in-person classes were conducted, focusing on Computational Thinking activities. We used the Developmental Channels Questionnaire, which includes 13 emotions rated on a Likert scale from 0 to 10, to measure emotions. The results showed that the predominant emotions were mostly positive and ambiguous, with low-intensity negative emotions, particularly in primary education. Gender differences were observed only in secondary education, while in primary education, the differences were not significant. Girls demonstrated an emotional evolution when engaging in these activities, unlike boys. These findings provide valuable quantitative insights for primary and secondary school teachers. Understanding the emotions experienced can help guide effective teaching approaches. By addressing emotional factors, educators can enhance students' interest in computer science, thus fostering a more inclusive and engaging learning environment.

Keywords: Computational Thinking, Emotions, Primary Education, Secondary Education

1. Introduction

The use of Information and Computer Technology is routine at every level of schooling. Over decades, professional and scientific computing societies have taken leading roles in providing support for higher

education in various ways, particularly in the formulation of curricular guidelines. The report of the last effort is called Computing Curricula 2020 (Force, 2020; Impagliazzo & Pears, 2018). It does not provide specific curricula for each computing discipline; instead, the report suggests and provides many opportunities. These include refreshing the paradigm of teaching and educating, moving from knowledge or outcomes to proficiencies, and engaging graduates to exploit the benefits of workplace competencies. The report does not address pre-baccalaureate education, although it occasionally mentions this area, specifying the extensive work done by the computing education community around the world to improve the availability and quality of computing-related courses in primary and secondary education, with a specific focus on improving the diversity of students who opt for careers in computing.

In pre-university education, some authors argue for the need to change the curricular guidelines of Computer Science by addressing the different key aspects on which they should focus, noting that all students should learn about them (Webb et al., 2017), or that the curricula should not be based on fashions and trends, but on contents and processes (Zendler et al., 2011). Moreover, Computer Science does not always pique the interest of young people since there is a lack of knowledge (Hubwieser et al., 2011) and there is the belief that it is complicated and beyond their reach (Giannakos et al., 2013). Also, recent studies show that popular stereotypes and identities of people who work with computers could potentially dissuade a pool of talents from contemplating computing careers as potential future pathways (Dou et al., 2020; Wong, 2016). Another consideration is the gender differences present in this field (Kim et al., 2021), with much fewer women than men in study fields that involve Computer Science (Strachan et al., 2018). One of the reasons for this lack of motivation is the stereotypes they have of computer scientists (Master et al., 2016).

Nowadays it is considered essential for anyone, in addition to having basic notions about computing, to also be knowledgeable of the operation of a programmable machine; that is, what can be automated and what cannot (Riesco et al., 2014). This could be addressed by working on Computational Thinking skills: the ability to solve problems, design systems, and understand human behavior through the use of essential concepts in Computer Science (Wing, 2006). It could also be described as those thought processes involved in formulating problems and representing their solutions, where said solutions can be executed by an information processing agent, be it a human, a computer, or a combination of the two. Some authors have also included in this definition a persistence in working with complicated problems or the ability to handle ambiguity (Barr & Stephenson, 2011). Others even go beyond computers, since it encompasses three areas, namely programming concepts (sequences, loops, events, etc.), certain practices that are developed through programming (improved problem-solving skills, repurposing, combining different projects, etc.) and perspectives from the world around us (expression, connecting with others, questioning ideas, etc.) (Brennan & Resnick, 2012). In addition, it has been shown that

Computational Thinking is an effective way by which it is possible to approach and increase the interest of girls in Computer Science (Seneviratne, 2017).

Some authors have supported the idea of introducing Computational Thinking in pre-university studies as a way of improving students' notion of Computer Science (Funke et al., 2016; Herrero-Álvarez et al., 2023; Herrero-Álvarez et al., Jan 2021; Herrero-Álvarez et al., Oct 2021; Lye & Koh, 2014). Also, some experiments have confirmed that giving students a course in which they practice programming through a gaming environment using a robot increases the prospects of providing effective programming education to elementary students (Shim et al., 2017), or that the students' inadequate background knowledge of this field could be improved by teaching programming to children and teenagers at schools (Resnick et al., 2009), but for this to happen, it is necessary to foster the dialogue between the communities of primary and higher education (Medeiros et al., 2019), in addition to training the relevant teachers in pre-university education (Kalogiannakis & Papadakis, 2017).

Since the aforementioned misconception could considerably reduce the interest in this academic field (Henry & Dumas, 2018), in this paper we selected a set of extracurricular activities designed to disseminate and promote Computer Science through the development of Computational Thinking skills. The main aim is to provide a methodology to introduce concepts related to Computational Thinking, and therefore to Computer Science. The Computational Thinking training phase of said methodology consists of a set of both plugged and unplugged Computational Thinking activities, which have been designed and scheduled in five sessions lasting four hours each, involving primary (8-9 years old) and secondary (12-13 years old) education students.

However, it is important to consider that emotions affect how we acquire knowledge, that is, how we learn (Pekrun, 1992; Weiner, 1984). Preschool students, those younger than 6 years old, have mostly positive attitudes and emotions towards science-related activities, but this positive predisposition decreases with age, especially between 8 and 10 years old (Dávila-Acedo et al., 2021; Mellado Jiménez et al., 2014; Osborne et al., 2003). Specifically, in the area of Computer Science, better results have been obtained when learning to program using a platform with systems that recognize emotions and adapt the content accordingly, than by using the same platform with the recognition system disabled (Zatarain Cabada et al., 2018). Other authors have conducted measurements involving the emotions felt when carrying out activities related to Computer Science, reaching the conclusion that happiness affected positively, and anxiety negatively (Giannakos et al., 2014). Furthermore, some studies point to a loss in efficiency due to anxiety in people who use computers (Achim & Kassim, 2015), or show that it is possible, by lowering anxiety and anger through computers, to improve one's knowledge of computers (Kay, 2008).

The purpose of this paper is to analyze the emotions that are present in young people who engage in Computational Thinking activities. This study contributes to the existing literature by examining the emotions associated with Computational Thinking, which is a distinct aspect of Computer Science education and is not widely studied. By categorizing these emotions as negative, ambiguous, and positive, we can explore their impact on individuals' perception of Computer Science, considering factors such as age and gender.

The rest of this paper is organized as follows. A further description of the hypotheses and research goals is given in the next subsection. Section 2 presents the methodology used in this study, with a description of both the activity sessions conducted and the measurement instruments utilized. In section 3, the results of the study are presented and discussed. Finally, section 4 contains the findings of our work and future areas of research.

1.1 Hypothesis, Aims, and Objectives

The hypothesis considered in this paper is that the poor interest in Computer Science shown by young people is due to their misconception about the field. We would also like to determine why the number of girls enrolled in engineering degrees is low (Strachan et al., 2018). Bearing the above in mind, Computational Thinking training would also allow girls to become much more interested in Computer Science. At this point, we should note that, due to the low participation of women in engineering degrees, the questionnaires and activities designed were analyzed from a gender perspective.

We aim to show that no gender differences exist in the emotional state when providing training on Computational Thinking in primary education, but that they do exist in secondary education. Also, recent studies have shown that girls tend to align with stereotypes related to subjects of a more verbal nature, while boys excel in Mathematics and Science. This difference occurs mainly in adolescence (Kurtz-Costes et al., 2014; Plante et al., 2009). In the case of university studies, and specifically in Computer Science studies, there are women who avoid difficult technical tasks for fear of affecting the team’s success, because of either their lack of experience or their lower self-efficacy in particular domains, influenced by gendered expectations of men’s experience (Fowler & Su, 2018). Therefore, it is important to approach the work from a gender perspective and ascertain why differences in the perception of Computer Science between genders are not expected in primary education, but are expected in secondary education. Another important aspect that can affect the emotions felt by students is related to the methodology of the activity they perform, guided or discovery, as well as the order in which they perform them (Goo et al., 2006), or even a combination of both, guided-discovery (Honomichl & Chen, 2012).

To promote pre-university education in the field of Science, Technology, Engineering, and Mathematics (STEM), it is necessary to train the relevant teachers, who must possess knowledge in this domain. However, they did not receive sufficient pre-service or in-service training, and lacked an adequate understanding of planning, implementing, and assessing activities (Gözüm et al., 2022).

Considering the aforementioned context, one of the main goals of this initiative is to make Computer Science much more appealing to young people through specific training on Computational Thinking. The main goal of this work is to analyze the emotional state of pre-university students as they engage in Computational Thinking activities, identifying what emotions are present during these sessions and their intensity and determine if there are differences in the emotions depending on the age, gender, and session model. The specific hypothesis are as follows:

H1: *Girls will feel fewer positive emotions than boys, especially in secondary school.*

H2: *Negative emotions will be higher in secondary school than in primary school.*

H3: *The session model does not significantly affect the emotions felt.*

2. Method

To develop Computational Thinking skills in young individuals, a course was conducted wherein primary and secondary students participated in a series of activities focused on exploring these concepts.

2.1 Activities

The students took a course with five lessons, two-hour classroom sessions, and a further 10 hours of homework. To train the students, a combination of both plugged and unplugged activities (that is, activities that rely on using a computer or mobile device, paper and pencil or any electronic device) and tools was used. It has been demonstrated that this type of activity effectively enhances interest in Computer Science among pre-university students (Herrero-Álvarez et al., 2023).

The activities were divided into two types, depending on the learning methodology. One was guided, where the basic concepts and principles of Computational Thinking were presented using a problem and analyzing the algorithm required to solve it. And the other involved discovery, where the student was taught the tools needed to implement some of the examples involved in Computational Thinking. The course employed two models: one began with two guided sessions and ended with three discovery sessions, guided-discovery (GD) model; in the other, the sessions were reversed, discovery-guided (DG) model.

These activities are described in Table 1 for both the primary and secondary levels. The activities for the DG model are the same but in different order, since the discovery activities are presented first, followed by the guided activities.

Table 1. Description of activities in the GD model

PRIMARY		
Guided		Discovery
Session 1	Session 2	Sessions 3, 4 & 5
Code&Go Mouse. Program a robot that travels in a maze[2].	Course at Code.org. Course 2[3].	Exercise on Scratch (Resnick et al., 2009). Fruit basket. The students program a basket in which they must place fruits without going over a specified calorie limit. Using the Makey Makey board[4], they cut out fruits from construction paper and line it with aluminum paper.
SECONDARY		
Guided		Discovery
Session 1	Sessions 2 & 3	Sessions 4 & 5
Course at Code.org. 20-hour course[5].	Exercise on Scratch. Matrioskas challenge. Arrange 5 dolls in size from smallest to largest. Discovery work continues in session 3	Robot mBot[6]. A self-steering robot with multiple sensors is programmed to travel in a circuit.

2.2 Participants

All the students participating in the project also participated in the study described in this work. This project was carried out with students in different schools on the island of Tenerife, Canary Islands, Spain, in 3rd grade, 8-9 years old, and 7th grade, 12-13 years old. Both girls and boys participated in

[2] https://www.learningresources.com/code-gor-robot-mouse-activity-set
[3] https://studio.code.org/s/course2
[4] https://makeymakey.com/
[5] https://studio.code.org/s/20-hour
[6] https://www.makeblock.com/mbot/

the study. The teaching staff responsible for them authorized their participation. No payment was made to the participants. All the data were collected in the schools that participated in the project, which was affected by the COVID-19 pandemic, meaning the expected sample size was reduced, as schools closed from March to June, especially the secondary education schools, which were scheduled for those months. In previous editions, the project was carried out with more than 250 students; however, in the school year that is the subject of this study (2019/2020), the total sample was 102 students. Table 2 lists all the students who took part in the project, grouped by grade and gender.

Table 2. Quantitative description of the sample

PRIMARY		**SECONDARY**	
74 students		28 students	
39 girls	35 boys	10 girls	18 boys

2.3 Data Collection

At the end of the first session, at the end of the first session after the methodology change, session 3 in the GD model and session 4 in the DG model, and in the last session, the participants completed the *Developmental Channels Questionnaire* - DCQ (Mosston & Ashworth, 2002), which was used to record their emotions. This questionnaire was available online and was completed by the students autonomously on the device they had used to carry out the different exercises of the project, which could be a computer or a tablet. This data collection method represents an affordable option with greater data completeness compared to data collection by paper (Ebert et al., 2018). This questionnaire also included a question about the student's gender (girl/boy).

Gathering data at the conclusion of each stage facilitates the examination of potential disparities that arise when implementing either methodology.

2.4 Instrumentation

The different methods of learning show the relationship between pedagogical elements by creating conditions for diverse experiences (Mosston & Ashworth, 2002), becoming a tool that teachers can use to express their creativity and individuality (Goldberger et al., 2012). The choice of teaching method is an important decision for instructors, since it affects their relationship with the various elements of the teaching activity (Tsolakidis & Anagnostou, 2011).

The DCQ includes scales described using opposing pairs of adjectives, such as minimum-maximum, hard-easy, strong-weak, bad-good, useful-useless, and pleasant-unpleasant, which provide an excellent gauge of an individual's thoughts. Specifically, it was used to ask about their happiness, compassion,

surprise, joy, sadness, fear, humor, anxiety, love, anger, rejection, shame, and hope, using a Likert scale from 0 to 10.

The students were given this questionnaire three times: at the end of the first session, at the end of the first session after the methodology change (session 3 in the GD model and session 4 in the DG model), and in the last session.

2.5 Data Analysis

After gathering the data from the questionnaires, the theoretical variables were calculated and classified into positive (happiness, joy, humor and love), negative (sadness, fear, anxiety, anger, rejection and shame), and ambiguous (compassion, surprise, hope), as per Lazarus (Lazarus, 1991) and Bisquerra (Bisquerra Alzina, 2003), based on the average score for each group of emotions. In Section 3 on the results, graphs are provided in a bar diagram format for each of group of emotions, separating them by gender and educational level, and by gender and type of session.

The data were analyzed using version 2.0 of the SPSS statistics program for Windows. The Kolmogorov-Smirnov normality tests show that the distributions of the theoretical variables do not follow a normal distribution, which translates into a lower reliability of the mean as a measure of central tendency. It is therefore possible that some of the trends observed as not significant are, nonetheless, sufficient to be considered important.

We then conducted a *Chi-square Automatic Interaction Detector* - CHAID - analysis (Kass, 1980), which yielded a representation of the data in decision trees for the Gender (girls or boys), Level (primary or secondary) and Session Type (GD or DG model) variables.

The data show an interrelation between the variables different from that suggested by Lazarus (Lazarus, 1991) and Bisquerra (Bisquerra Alzina, 2003). Preliminary tests using exploratory factor analysis demonstrate a two-factor result, positive + ambiguous and negative, instead of three. In future work, the results could gain in strength by considering only these two factors.

3. Results

In this section, the results of the *DCQ* questionnaire are presented, analyzing them first according to gender and educational level, then according to gender and session model, and finally the decision trees are included according to the *CHAID* analysis (Kass, 1980) for each classification of emotions: positive, negative and ambiguous.

3.1 Emotions, gender and educational level

This section specifies the positive, ambiguous and negative emotions obtained depending on the students' educational level. Figure 1 shows the positive, ambiguous and negative emotions by gender and educational level, reflecting the median for each classification of emotions on a scale from 0 to 10.

As we can see, there are no apparent differences between boys and girls at the primary level; however, the girls in secondary school express fewer positive emotions, with a difference of more than one point.

As regards the ambiguous emotions by gender and level of education, there are differences between boys and girls in both educational levels, although both girls and boys feel fewer ambiguous emotions in secondary school. Both tendencies, boys and girls, seem to be present in equal measure, but as with the positive emotions, the difference is greater in secondary school, where girls feel somewhat less ambiguous emotions, with a difference of up to one point on average.

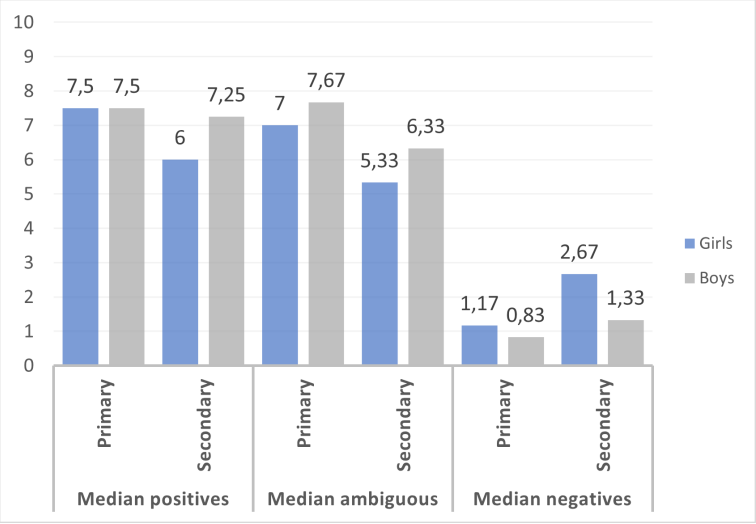


Figure 1. Emotions by gender and educational level

Even though the negative emotions in both levels are very low -close to zero-, we see significant differences between the levels, these differences being much starker in the girls than in the boys. The girls in secondary school stand out, where the greatest number of negative emotions was evident when carrying out the activities, near three points out of 10. These differences are much more significant in the girls than in the boys, which reaffirms hypothesis **H1**. In general, negative emotions are more present in secondary school than in primary school, so hypothesis **H2** is accepted.

3.2 Emotions, gender and session model

This section presents the median of the positive, neutral and negative emotions obtained based on the session model employed. Figure 2 shows the positive, neutral and negative emotions based on the gender and session model employed.

We see differences between the boys and girls in the Discovery-Guided sessions, but there seem to be no differences between them in the Guided-Discovery sessions. Moreover, the Discovery-Guided sessions seem to provoke a lower number of positive emotions in the girls, of one point out of 10. We see no large differences between the boys and girls regarding ambiguous emotions; however, there is a slight change in the boys during the Discovery-Guided sessions, since there is a difference of about one point higher with respect to the girls.

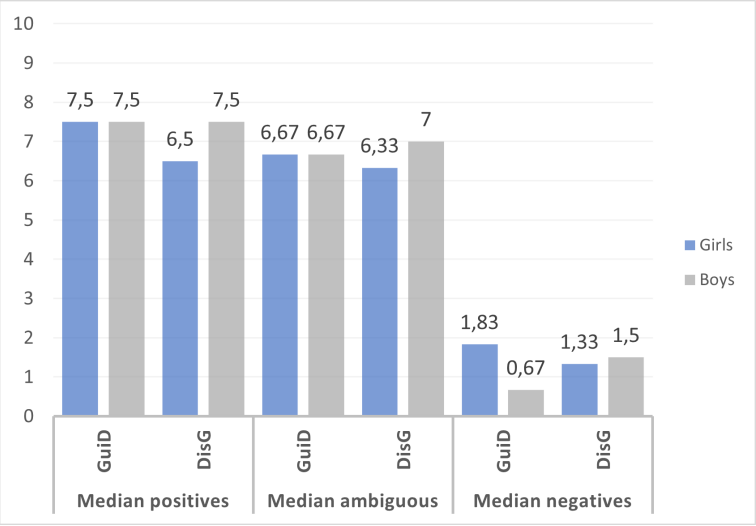


Figure 2. Emotions by gender and session model

As for the negative emotions by gender and session model, there are no apparent significant differences between the two models. What is more, we see an inverse relationship in the differences between the boys and the girls, such that the girls seem to develop more negative emotions during the Guided-Discovery sessions, whereas the boys develop more negative emotions during the Discovery-Guided sessions. Despite the differences observed, these are not significant, so hypothesis **H3** is accepted.

3.3 Emotions, gender and educational level model trees

This section presents the classification trees for the positive, ambiguous and negative emotions obtained depending on educational level and gender, as shown in the next three figures.

These decision trees contain different nodes showing the number of sample data for that node '*n*', the mean score '*mean*', the standard deviation '*Std. Dev.*', and the percentage of the total sample that this node represents '*%*', as per the CHAID analysis method (Kass, 1980). The *n* in the trees indicates the total number of tests collected, considering that this was completed by each student three times for each session and that data cleaning was not performed for this analysis.

Each level of the decision trees contains the statistical analysis performed, such that the next nodes of the next level are those where the greatest differences are evident, where the p-value < 0.05.

3.3.1 Tree of positive emotions

The decision tree for positive emotions, see Figure 3, shows that the most significant difference is found between secondary school students, with boys feeling more positive emotions than girls, so hypothesis **H1** is accepted. The values obtained were p-value $= 0.017$; $\mathrm{F} = 5.839$. There are also differences between the educational level, with the younger students feeling these kinds of emotions more than the secondary students.

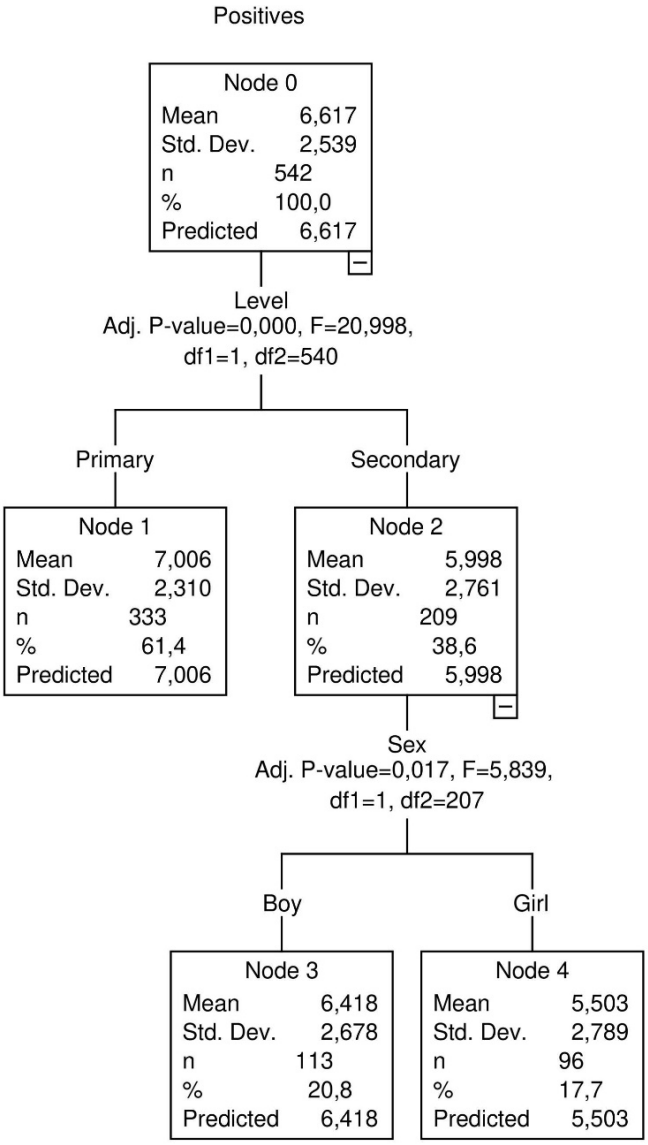


Figure 3. Tree of positive emotions

3.3.2 Tree of ambiguous emotions

Again, in the decision tree of ambiguous emotions, see Figure 4, we see that the most significant difference is found between the older boys and girls, since the latter feel these types of emotions to a lesser extent. The values obtained were p-value $= 0.014$; $F = 6{,}188$. Regarding the educational level, differences also appear, since in the case of primary school, these types of emotions are greater than in secondary school, with a difference of more than 1.5 points.

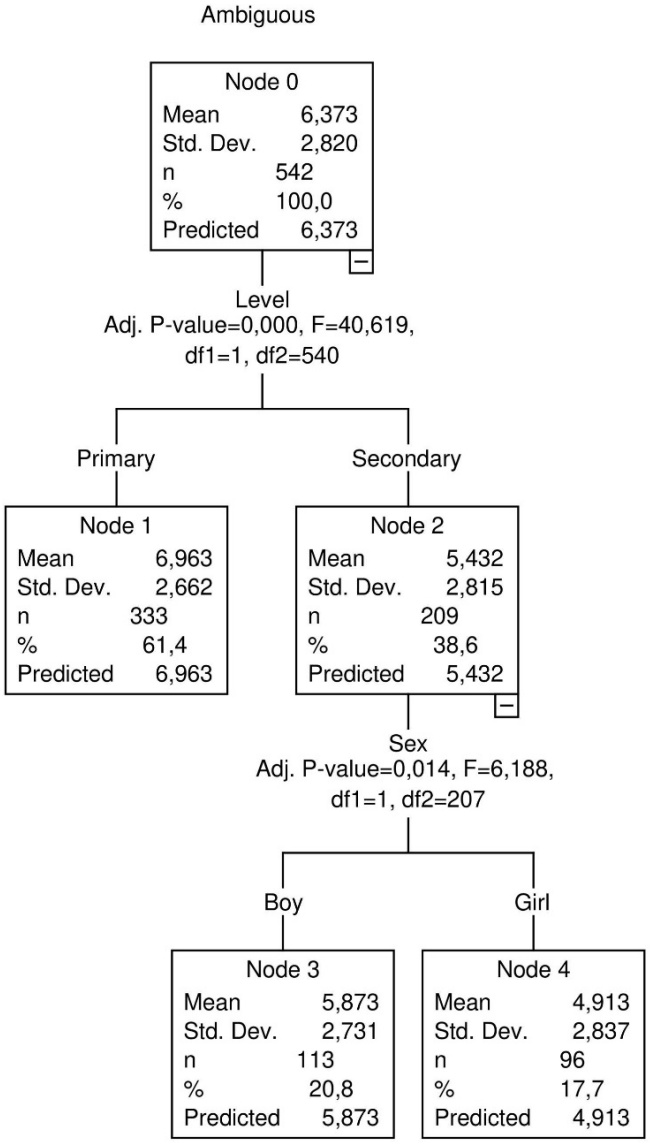


Figure 4. Tree of ambiguous emotions

3.3.3 Tree of negative emotions

In the case of the decision tree of negative emotions, see Figure 5, the most significant differences are again apparent in secondary school between girls and boys, with girls being the ones who feel the most negative emotions when engaged in activities involving Computational Thinking. The values obtained were p-value $= 0.014$; $F = 6{,}207$. The primary school students feel fewer negative emotions than the secondary school students, since the former do not reach two points, while the latter exceed 2.5 points out of 10, so hypothesis **H2** is accepted.

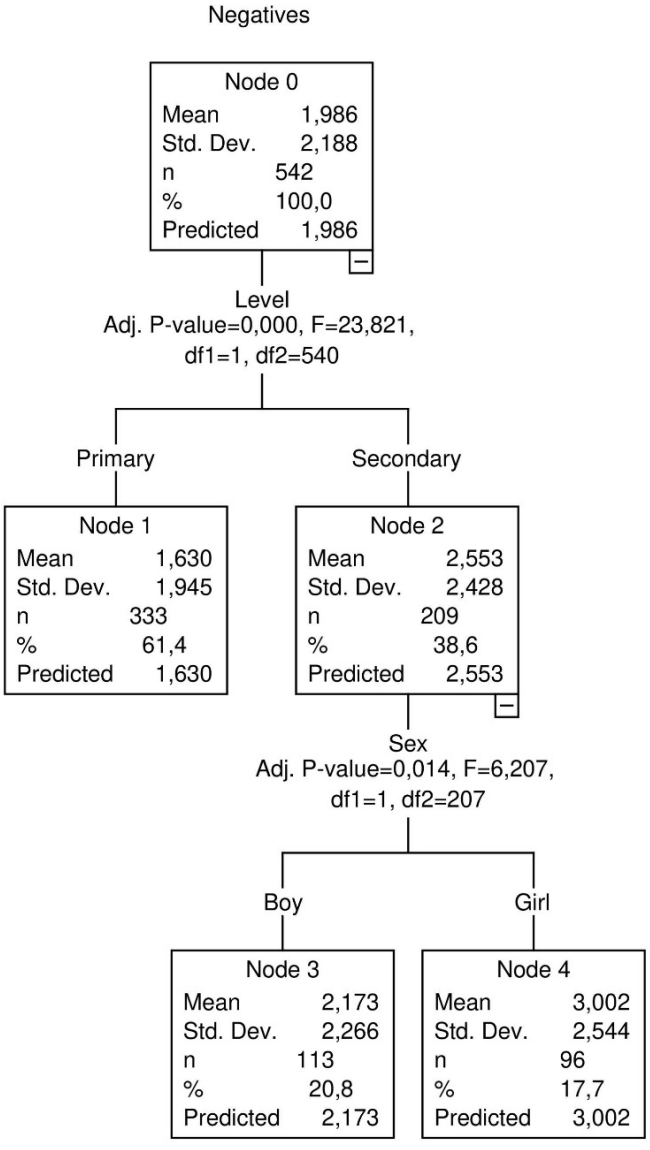


Figure 5. Tree of negative emotions

4. Conclusions and discussion

The main conclusions of this study have been established based on the objectives and hypotheses proposed for its development. Regarding the objective of identifying the emotions that are produced, as

well as their intensity, we conclude that positive and ambiguous emotions are mainly produced in this type of session with intensity values of 6.62 and 6.37 on average, respectively. Although the students show negative emotions with low intensity, with an average of 1.99 over 10.0, it is in secondary education where they mostly appear.

In addition, regarding the study of the possible differences between the emotions felt depending on the session model, Guided-Discovery or Discovery-Guided, we conclude that at the primary and secondary educational level, it follows that, with the data collected, similar positive, negative, and ambiguous emotions can be found between boys and girls without finding significant differences between the two genders in any of them, but there is an inverse relationship in the differences between the boys and girls, since the girls seem to develop more negative emotions during the Guided-Discovery sessions, but the boys develop more negative emotions during the Discovery-Guided sessions.

However, regarding the educational level, primary or secondary, it is necessary to note that in secondary, there are differences between boys and girls in all of them. We see that the girls exhibit more important changes in this type of session, whereas the boys hardly evolve emotionally in this aspect. Thus, the girls seem to show a lower intensity of positive and ambiguous emotions, and a slight increase of negative ones.

Our results show that the hypotheses considered at the beginning of the work are accepted. In addition, they are consistent with what other authors have confirmed regarding the change that occurs in emotions with age, where positive emotions decrease, and regarding how in the case of girls, there is a more noticeable difference with respect to the boys as they grow up.

The implications of this work, which observes the emotions felt by the students when carrying out these activities, mean that by knowing what the students feel, it is possible to adapt the activities proposed so that they are more appealing to the students, improving their learning process, guiding them to those that produce the greatest number of positive emotions, or eliminating those that produce negative ones. In this work, we have observed that the positive emotions in primary school are greater, changing completely when reaching secondary education, so it would be interesting to adapt the activities before reaching this educational stage. It is thus essential to work on maintaining positive emotions as the students grow up in order to keep their interest, especially in girls, since many of the studies focus on the emotions experienced by students regarding Computer Science, either through computer usage or by engaging in specific activities, without addressing the training of Computational Thinking skills. This work offers a review that can assist pre-university teachers with guiding various exercises aimed at enhancing students' emotional response.

The results show that the highest number of negative emotions is observed among female students in secondary education. Therefore, a special effort should be made to conduct activities that foster interest at these ages. One of the future objectives is to conduct a detailed study of the emotional response to each activity, specifically regarding programming concepts, regardless of whether they were taught through guided or discovery-based approaches.

Acknowledgments

The work of Rafael Herrero-Álvarez has been financially supported by Gobierno de Canarias through the Agencia Canaria de Investigación, Innovación y Sociedad de la Información - ACIISI - with the contract TESIS2021010058.

This work also has been partially funded by the project ``Piens@ Computacion@ULLmente (REF~A22120132). Programa educativo para el fomento del pensamiento computacional a través de la realización de actividades que permitan su desarrollo y su inclusión en el currículo" of Cabildo de Tenerife and Fundación General de la Universidad de La Laguna.

References

Achim, N., & Kassim, A. A. (2015). Computer Usage: The Impact of Computer Anxiety and Computer Self-efficacy. *Procedia, Social and Behavioral Sciences, 172*, 701-708. https://10.1016/j.sbspro.2015.01.422

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads, 2*(1), 48-54. https://10.1145/1929887.1929905

Bisquerra Alzina, R. (2003). Educación emocional y competencias básicas para la vida. *Revista De Investigación Educativa, 21*(1), 7-43.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at the *Annual American Educational Research Association Meeting,*

Dávila-Acedo, M. A., Airado-Rodríguez, D., Cañada-Cañada, F., & Sánchez-Martín, J. (2021). Detailed Emotional Profile of Secondary Education Students Toward Learning Physics and Chemistry. *Frontiers in Psychology, 12*, 659009. https://10.3389/fpsyg.2021.659009

Dou, R., Bhutta, K., Ross, M., Kramer, L., & Thamotharan, V. (2020). The Effects of Computer Science Stereotypes and Interest on Middle School Boys' Career Intentions. *ACM Transactions on Computing Education, 20*(3), 1-15. https://10.1145/3394964

Ebert, J. F., Huibers, L., Christensen, B., & Christensen, M. B. (2018). Paper- or Web-Based Questionnaire Invitations as a Method for Data Collection: Cross-Sectional Comparative Study of Differences in Response Rate, Completeness of Data, and Financial Cost. *Journal of Medical Internet Research, 20*(1), e24. https://10.2196/jmir.8353

Force, C. T. (2020). *Computing Curricula 2020: Paradigms for Global Computing Education (CC2020)* (2020 December 31 ed.). Association for Computing Machinery. https://10.1145/3456302

Fowler, R. R., & Su, M. P. (2018). Gendered Risks of Team-Based Learning: A Model of Inequitable Task Allocation in Project-Based Learning. *IEEE Transactions on Education, 61*(4), 312-318. https://10.1109/TE.2018.2816010

Funke, A., Berges, M., & Hubwieser, P. (2016). Different Perceptions of Computer Science. Paper presented at the *International Conference on Learning and Teaching in Computing and Engineering (LaTICE),* 14-18. https://10.1109/LaTiCE.2016.1

Giannakos, M. N., Jaccheri, L., & Leftheriotis, I. (2014). Happy Girls Engaging with Technology: Assessing Emotions and Engagement Related to Programming Activities. (pp. 398-409). Springer International Publishing. https://10.1007/978-3-319-07482-5_38

Giannakos, M. N., Jaccheri, L., & Proto, R. (2013). Teaching Computer Science to Young Children through Creativity: Lessons Learned from the Case of Norway. Paper presented at the *Computer Science Education Research Conference,* 103-111.

Goldberger, M., Ashworth, S., & Byra, M. (2012). Spectrum of Teaching Styles Retrospective 2012. *Quest (National Association for Kinesiology in Higher Education), 64*(4), 268-282. https://10.1080/00336297.2012.706883

Goo, J. J., Park, K. S., Lee, M., Park, J., Hahn, M., Ahn, H., & Picard, R. W. (2006). Effects of Guided and Unguided Style Learning on User Attention in a Virtual Environment. *Lecture notes in computer science* (pp. 1208-1222). Springer Berlin Heidelberg. https://10.1007/11736639_151

Gözüm, A. İ C., Papadakis, S., & Kalogiannakis, M. (2022). Preschool teachers' STEM pedagogical content knowledge: A comparative study of teachers in Greece and Turkey. *Frontiers in Psychology, 13*, 996338. https://10.3389/fpsyg.2022.996338

Henry, J., & Dumas, B. (2018). Perceptions of computer science among children after a hands-on activity: A pilot study. Paper presented at the *2018 IEEE Global Engineering Education Conference (EDUCON),* 1811-1817. https://10.1109/EDUCON.2018.8363454

Herrero-Álvarez, R., León, C., Miranda, G., & Segredo, E. (Jan 2021). Propuesta de actividades para el desarrollo del pensamiento computacional en estudios pre-universitarios . Paper presented at the *XI Congreso Iberoamericano De Docencia Universitaria (CIDU 2020),* 306-3019.

Herrero-Álvarez, R., León, C., Miranda, G., Segredo, E., Socas, Ó, García, L., & Díaz, Y. (Oct 2021). Metodología para el desarrollo del pensamiento computacional en tiempos de COVID-19. Paper presented at the *VI Congreso Internacional Sobre Aprendizaje, Innovación Y Cooperación (CINAIC 2021),* https://10.26754/CINAIC.2021.0051

Herrero-Álvarez, R., Miranda, G., León, C., & Segredo, E. (2023). Engaging Primary and Secondary School Students in Computer Science through Computational Thinking Training. *IEEE Transactions on Emerging Topics in Computing, 11*(1), 56-69. https://10.1109/TETC.2022.3163650

Honomichl, R. D., & Chen, Z. (2012). The role of guidance in children's discovery learning. *Wiley Interdisciplinary Reviews. Cognitive Science, 3*(6), 615-622. https://10.1002/wcs.1199

Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M., Knobelsdorf, M., Magenheim, J., Mittermeir, R., & Schubert, S. (2011). Computer science/informatics in secondary education. *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education - Working Group Reports,* , 19-38. https://10.1145/2078856.2078859

Impagliazzo, J., & Pears, A. N. (2018). The CC2020 project—computing curricula guidelines for the 2020s. Paper presented at the *2018 IEEE Global Engineering Education Conference (EDUCON),* 2021-2024.

Kalogiannakis, M., & Papadakis, S. (2017). A proposal for teaching ScratchJr programming environment in preservice kindergarten teachers. Paper presented at the *12th Conference of the European Science Education Research Association (ESERA),*

Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Journal of the Royal Statistical Society.Series C (Applied Statistics), 29*(2), 119-127. https://10.2307/2986296

Kay, R. H. (2008). Exploring the relationship between emotions and the acquisition of computer knowledge. *Computers and Education, 50*(4), 1269-1283. https://10.1016/j.compedu.2006.12.002

Kim, H. S., Kim, S., & Lee, W. J. (2021). Extending Computational Thinking into Information and Communication Technology Literacy Measurement. *ACM Transactions on Computing Education, 21*(1), 1-25. https://10.1145/3427596

Kurtz-Costes, B., Copping, K. E., Rowley, S. J., & Kinlaw, C. R. (2014). Gender and age differences in awareness and endorsement of gender stereotypes about academic abilities. *European Journal of Psychology of Education, 29*(4), 603-618. https://10.1007/s10212-014-0216-7

Lazarus, R. S. (1991). *Emotion and Adaptation*. Oxford University Press.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://10.1016/j.chb.2014.09.012

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing Whether She Belongs: Stereotypes Undermine Girls' Interest and Sense of Belonging in Computer Science. *Journal of Educational Psychology, 108*(3), 424-437. https://10.1037/edu0000061

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education, 62*(2), 77-90. https://10.1109/TE.2018.2864133

Mellado Jiménez, V., Borrachero, A. B., Brígido, M., Melo, L. V., Dávila, M. A., Cañada, F., & ., E. a. (2014). Emotions in Science teaching. *Enseñanza De Las Ciencias, 32*(3), 11-36. https://10.5565/rev/ensciencias.1478

Mosston, M., & Ashworth, S. (2002). *Teaching Physical Education* (Fifth ed.). Benjamin Cummings.

Osborne, J., Simon, S., & Collins, S. (2003). Attitudes towards science: a review of the literature and its implications. *International Journal of Science Education, 25*(9), 1049-1079. https://10.1080/0950069032000032199

Pekrun, R. (1992). The impact of emotions on learning and achievement: towards a theory of cognitive/motivational mediators. *Applied Psychology, 41*(4), 359-376.

Plante, I., Théorêt, M., & Favreau, O. E. (2009). Student gender stereotypes: contrasting the perceived maleness and femaleness of mathematics and language. *Educational Psychology (Dorchester-on-Thames), 29*(4), 385-405. https://10.1080/01443410902971500

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All. *Commun.ACM, 52*(11), 60-67. https://10.1145/1592761.1592779

Riesco, M., Fondón, M., Gutiérrez, D. Á, Lopez, B., Cernuda, A., & Juan, A. (2014). La Informática como materia fundamental en un sistema educativo del siglo XXI. Paper presented at the *Xx Jenui,* 27-32.

Seneviratne, O. (2017). Making Computer Science Attractive to High School Girls with Computational Thinking Approaches: A Case Study. *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 21-32). Springer International Publishing. https://10.1007/978-3-319-52691-1_2

Shim, J., Kwon, D., & Lee, W. (2017). The Effects of a Robot Game Environment on Computer Programming Education for Elementary School Students. *IEEE Transactions on Education, 60*(2), 164-172. https://10.1109/TE.2016.2622227

Strachan, R., Peixoto, A., Emembolu, I., & Restivo, M. T. (2018). Women in engineering: Addressing the gender gap, exploring trust and our unconscious bias. *2018 IEEE Global Engineering Education Conference (EDUCON),* , 2088-2093. https://10.1109/EDUCON.2018.8363497

Tsolakidis, S., & Anagnostou, G. (2011). The impact of physical education teaching styles on the construction of pupils' literate subjectivities. Paper presented at the *Proceedings of the 31st Annual Meeting of the Department of Linguistics of the Faculty of Philosophy,* 145-154.

Webb, M., Davis, N., Bell, T., Katz, Y., Reynolds, N., Chambers, D., & Sysło, M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies, 22*(2), 445-468. https://10.1007/s10639-016-9493-x

Weiner, B. (1984). Attributional Theory of Motivation and Emotion. *Psychological Review,* https://10.1007/978-1-4612-4948-1

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. https://10.1145/1118178.1118215

Wong, B. (2016). 'I'm good, but not that good': digitally-skilled young people's identity in computing. *Computer Science Education, 26*(4), 299-317. https://10.1080/08993408.2017.1292604

Zatarain Cabada, R., María Lucía Barrón Estrada, José Mario Ríos Félix, & Giner Alor Hernández. (2018). A virtual environment for learning computer coding using gamification and emotion recognition. *Interactive Learning Environments, 28*(8), 1048-1063. https://10.1080/10494820.2018.1558256

Zendler, A., Spannagel, C., & Klaudt, D. (2011). Marrying Content and Process in Computer Science Education. *IEEE Transactions on Education, 54*(3), 387-397. https://10.1109/TE.2010.2062184

An Investigation of In-service Teachers' Perceptions and Development of Computational Thinking Skills in a Graduate Emerging Technologies Course

Yi JIN[1]

Jason R. HARRON[1]

[1]Kennesaw State University, United States of America

DOI: https://doi.org/10.21585/ijcses.v6i2.165

Abstract

Computer science (CS) has become a critical part of K–12 education worldwide. Computational thinking (CT) skills are a key set of competencies in CS education that can solve problems and use computational design to create useful solutions. However, preservice and in-service teachers are not fully prepared to integrate CS and CT into their curricula. Furthermore, there are limited special topic courses and educational research on how to facilitate in-service teachers' professional learning of CS and CT, as well as their content-specific integration. Therefore, this study investigated in-service teachers' perceptions and development of CT skills in an online graduate emerging technologies course. Theoretically framed by the four cornerstones of CT (i.e., abstraction, algorithms, decomposition, and pattern recognition), participants perceived that they increased their CT problem-solving and creativity skills but decreased their collaborative learning and critical thinking skills. Additionally, teachers increased their CT test scores after taking the course. Most teachers used CT terminology correctly (i.e., algorithms and decomposition). However, only 59% correctly described abstraction and pattern recognition, while most teachers did not mention debugging. The authors call on teacher educators to address in-service teachers' CS knowledge gaps, increase their CT skills, and select appropriate strategies for CT professional learning.

Keywords: computational thinking, creative computing, online learning, perceptions, teacher education

K. Introduction

Computational Thinking (CT) skills are a key set of competencies that combine problem-solving and computational design to create useful solutions (Grover & Pea, 2018). Students and teachers with CT

skills can collect and analyze data, decompose problems, recognize patterns, and filter out variables to find novel and elegant solutions. CT helps people to think like computer scientists and transform complex problems into ones that can be easily understood across a wide range of subjects. In combination, CT and coding have immense potential to transform K–12 education by integrating core computational concepts and principles across the curriculum.

In recent years, movements at the national and state levels in the U.S. have aimed to introduce students to computer science (CS) education by establishing frameworks, standards, and curricula with the goal of expanding CS opportunities to all. Nationally, this push includes the development of the *K–12 Computer Science Framework* (2016), which highlights CT as one of four significant themes that are interwoven throughout. This framework aligns with the *International Society for Technology in Education (ISTE) Standards for Educators* and *Students* by sharing the vision that CT is important for all teachers and students (ISTE, 2016a, 2016b). Based on these efforts, the Computer Science Teachers Association (CSTA) has proposed a comprehensive set of K–12 standards in collaboration with multiple national and international associations to guide how CS education is implemented in practice (CSTA, 2017). Similarly, many countries have incorporated CS education into their curriculum (Dufva & Dufva, 2016).

Due to these collective endeavors, CSforALL movements have been fruitful in the U.S. According to the *2022 State of Computer Science Education* report, 37 states have adopted at least five of nine recommended policies to make CS part of the education system while 27 states require all high schools to offer at least one CS course (Code.org, CSTA, & ECEP Alliance, 2022). Across the U.S., 53% of public high schools (13,865) offer fundamental CS, up from 35% in 2018. Moreover, 76% of students attend a high school that offers a foundational CS course. All 50 states and Washington D.C. allow CS courses to be counted toward the graduation requirement. Furthermore, Arkansas, Nebraska, Nevada, South Carolina, and Tennessee require high school students to take CS courses for graduation. Although there are great advances in offering CS courses at the high school level, only 3.9% of middle school and 7.3% of elementary school students from the 19 states who reported middle and elementary school data offered foundational CS in grades K-8, highlighting the need to integrate CS into all content areas at the K-8 level to broaden participation (Code.org, CSTA, & ECEP Alliance, 2022; Kennedy et al., 2021).

Despite the growth in CS offerings, there continue to be access issues in K–12. First, access disparities persist in rural schools, urban schools, and schools with high percentages of economically disadvantaged students. These disparities also exist across gender boundaries, with fewer female students enrolled in CS courses across the elementary (49%), middle (44%), and high school (32%) grade bands (Code.org, CSTA, & ECEP Alliance, 2022). Furthermore, students from underrepresented populations, such as African American, Hispanic/Latino/Latina/Latinx, and Native American/Alaskan, are less likely to have

CS courses offered at their schools. Compared to their white and Asian peers, Hispanic/Latino/Latina/Latinx high school students are 1.4 times less likely to take a CS course. Similarly, English language learners, students with disabilities, and economically disadvantaged students are underrepresented in CS courses. These data emphasize that besides learning about CS and CT, preservice and in-service teachers also need to proactively seek strategies to teach these underrepresented students.

Although there are strong pleas to integrate CS and CT into all K–12 content areas (Grover & Pea, 2018; Kennedy et al., 2021), most teachers have not been able to achieve this goal in practice. One significant barrier causing the stagnant CT implementation includes a lack of preparation from teacher education programs and minimum professional development from schools and districts. For example, research shows that few teacher education programs provide CT training to preservice teachers (Yadav et al., 2017a). In addition, many K–12 in-service teachers had little knowledge about CT and did not know how to implement CT in their classrooms (Sands et al., 2018). In-service teachers also lack strategies for teaching CS and CT to underrepresented students (Gretter et al., 2019). Teachers even expressed that they were anxious about developing new learning resources and using novel technologies (Meerbaum-Salant et al., 2013), especially when teaching CT concepts and computing-related subjects (Grover & Pea, 2013). All these shortcomings underline the need for teacher educators to provide support and professional learning to both preservice and in-service teachers in integrating CS and CT into their subject areas and curricula (Voogt et al., 2015; Yadav et al., 2017b).

For in-service teachers, research has shown that targeted professional learning helps teachers improve their CT understanding and skills (Bower et al., 2017; Jaipal-Jamani & Angeli, 2017; Ketelhut et al., 2020). However, professional learning in literature occurred mostly in professional development programs, not courses in teacher education. Therefore, educational researchers need to design specific courses that facilitate teachers' professional learning in CS and CT, especially for elementary and middle school in-service teachers to design content-specific integration (Kennedy et al., 2021). In turn, this need warrants more studies examining the effectiveness of such courses. There is a limited number of this type of research in literature, especially those focusing on using the creative coding concept (Brennan, 2015; Yurkofsky et al., 2019). Thus, this study aims to investigate in-service teachers' perceptions and development of CT skills in a required emerging technologies course as part of an online instructional technology graduate program. The details of the design of this professional learning course and its effectiveness shed light on how to prepare in-service teachers to integrate CS and CT into their content areas. Moreover, the findings add to the literature on CT integration using the creative coding concept. Therefore, the current research intends to answer the following research questions:

(1) What are in-service teachers' perceptions about their CT skills before and after taking the graduate emerging technologies course?
(2) Is there a difference in in-service teachers' CT test scores after taking the course?
(3) How frequently and accurately do in-service teachers apply CT terminology in their final reports?

2. Literature Review

To better understand what researchers currently know about how teachers develop their CT skills, a review of the literature is provided below. This review includes a brief overview of the skills, practices, and pedagogy associated with CT, and summarizes how CT has been studied in K–12 and teacher education programs.

2.1 Computational Thinking Skills, Practices, and Pedagogy

Computational thinking (CT) has its origins in the 1980s, stemming from research about using personal computers and computing environments to support the social processes of learning while aiding in the development of higher-order cognitive skills (Papert, 1980; Pea & Kurland, 1984; Solomon, 1988). Wing (2006) brought CT to the mainstream discussion with her seminal and influential *Communications of the ACM* article, where she argues that CT is not only for computer scientists but serves as a set of attitudes and skills that are universally applicable to everyone. In particular, CT provides its users with various mental tools to solve problems, design systems, and understand human behaviors using a broad range of CS concepts.

Since the publication of Wing's article over 15 years ago, there have been more than 31,000 publications about CT indexed by Google Scholar. Expanding upon Wing's foundational definition, Barr and Stephenson (2011) provided educators with an operational definition, which defined CT as a problem-solving process involving the following steps: (a) formulating a problem in such a way that the use of computer technology can help us solve it; (b) analyzing data and representing that data through models or simulations; (c) identifying possible solutions to the problem posed; (d) generalizing this process to a wide variety of situations and issues.

However, despite the popularity of CT within the educational research community, there is still no consensus about how CT should be universally defined (Cansu & Cansu, 2019; Grover & Pea, 2018). The early definitions, which centered around the four cornerstones of abstraction, algorithms, decomposition, and pattern recognition, have been expanded upon to include a wide variety of CT skills/concepts and practices. For example, Mills et al. (2021) recently published a report that places CT

at the intersection of computing, computer science, and programming. Their report proposes that CT consists of a set of skills and practices that can be applied to solve problems. CT skills include abstraction, algorithmic thinking, debugging, decomposition, pattern recognition, and selecting tools. CT practices combine these skills to solve problems through the creation of computer programs (i.e., automation), data visualizations, or computational models. Lastly, these CT skills and practices are centered around the use of inclusive pedagogies which includes strategies "for engaging all learners in computing, connecting applications to students' interests and experiences, and providing opportunities to acknowledge and combat biases and stereotypes within the computing field" (Mills et al., 2021, p. 10).

Similarly, Yaşar et al. (2015) considered computational pedagogy an inherent outcome of computing, math, science, and technology integration. They firmly believe that computational 96odelling and simulation technology (CMST) can be used to improve teachers' technological pedagogical content knowledge (TPACK) (Mishra & Koehler, 2006; Yaşar et al., 2015). Thus, Yaşar et al. (2015) extended TPACK into Computational Pedagogical Content Knowledge to highlight computational pedagogy.

For this particular study, the researchers decided to use the operational definitions from the BBC Bitesize courses, which were also used as instructional materials in the course. The website defines that "computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand" (BBC Bitesize, n.d., What is computational thinking section, para. 2). Furthermore, they define the four cornerstones of CT as

- Decomposition — Breaking down a complex problem or system into smaller, more manageable parts.
- Pattern recognition — Looking for similarities among and within problems.
- Abstraction — Focusing on the important information only, ignoring irrelevant detail.
- Algorithms — Developing a step-by-step solution to the problem, or the rules to follow to solve the problem (BBC Bitesize, n.d., What is computational thinking section, para. 3).

2.2 Computational Thinking in K–12 Education

Traditionally, CS has been introduced at the high school level, focusing on teaching the computer programming skills needed to pass the AP CS exam (Goode, 2008). CT breaks this mold by acknowledging that students in younger grades (K–3) have the cognitive capabilities to apply computational skills to relevant problems (Papdakis, 2021; 2022). These skills can be introduced through "unplugged" activities that do not require digital devices (Mills et al., 2021), such as having

students give each other step-by-step instructions on how to brush their teeth (Hello Ruby, 2019). Other developmentally appropriate devices, such as Beebots or Codeapillar, allow students to manually program algorithms by giving step-by-step instructions at the push of a button (Papadakis et al., 2021). Besides these physical computing tools and activities, coding apps are used widely by younger learners, such as ScratchJr, Lightbot, Kodable, and Daisy the Dinosaur (Papdakis, 2021). In particular, Papadakis (2022) conducted a literature review on ScratchJr and found that it helped young learners understand CT concepts, practice coding skills, develop social-emotional skills, introduce students to STEM learning, especially numeracy concepts, and help them develop problem-solving strategies, planning methods, and thinking skills. Therefore, CT can be taught to young students and should be taught as early as possible (Kotsopoulos et al., 2017; Papadakis, 2021; 2022; Yadav et al., 2011).

In upper-grade levels (4–12), students can continue to develop their CT skills through the use of block-based programming languages, such as Scratch, or through the exploration of devices that utilize the Blockly programming library (Weintrop, 2021). Some of these devices include BBC micro:bit, Circuit Playground Express, Lego Mindstorms, Ozobots, Raspberry Pi, and Sphero. The user-friendly nature of these block-based programming languages allows for an entry point to computer science not only for students but also for teachers who are learning to code for the first time. Kalogiannakis et al. (2021) conducted a systematic review of the use of BBC micro:bit in elementary schools. They found that students and teachers show a positive attitude towards the tool. Moreover, students believe that micro:bit encourages creativity and facilitates their learning of the conceptual and procedural knowledge of CT and problem-solving. However, the findings also indicate teachers' lack of confidence in designing their own activities and instructions.

There is a trend to integrate CT into K–12 content areas. For example, CT has become a core scientific practice in STEM (NGSS, 2013; Weintrop et al., 2016). To facilitate empirical research, Weintrop et al. (2016) proposed a Computational Thinking in Mathematics and Science Taxonomy with four categories to ground CT in STEM. These categories include (a) data practices, (b) 97odelling and simulation practices, (c) computational problem-solving practices, and (d) systems thinking practices. Furthermore, CT integration into the science classrooms is well-researched on topics such as adding coding activities with little support for science learning (Grover et al., 2015), integrating CT into the science content knowledge of science textbooks (Wilkerson & Fenwick, 2017), and integrating computation as used by STEM professionals (Orton et al., 2016).

Empirical research about CT integration in math is expanding as well. In a scoping review, Hicknott et al. (2017) found that most CT integration in K–12 mathematics classrooms mainly concentrated on teaching programming skills and rarely focused on mathematical concepts in probability, statistics, and measurement of functions. Likewise, Barcelos et al. (2018) conducted a systematic review and found

42 studies. Fourteen programming languages were used in 22 studies, with Scratch being the most popular one. These studies also covered a wide range of math skills and contents, which were developed in conjunction with CT. The researchers suggested that interest in investigating the relationship between CT and math was growing.

Concerning CT instructions in K–12 settings, two main approaches are used, unplugged and programming activities. Huang and Looi (2021) conducted a critical review of the unplugged pedagogies used in K–12. They found that most unplugged activities were designed for younger students and non-specialist teachers and they were popular across age groups and learner characteristics. They summarized that unplugged pedagogy supports CT development, complements programming to develop CT, integrates with other subjects to develop CT, and facilitates teacher learning about CT and CS.

For teaching coding in K–12, Hsu et al. (2018) found that teachers mostly used visual programming languages in their CT instruction. Teachers' top strategies for CT instruction are project-based learning, problem-based learning, cooperative learning, and game-based learning. In contrast, other activities involving aesthetic experience, design-based learning, and storytelling are rarely adopted. To determine the general effectiveness of using programming for developing K–12 students' CT skills, Sun et al. (2021) conducted a meta-analysis. They found 86 empirical studies with 114 effect sizes. According to their results, programming activities could improve K–12 students' CT skills. They also found some instructional design factors that were more conducive to the goal, which were interdisciplinary integration of programming, setting the duration to be within one week to one month, having a class size of fewer than 50 students, and a practical selection of programming instrument and CT assessment types. Because of the popularity of Scratch as a programming language in K–12 CT instruction, numerous scholars have conducted research to analyze the impact of Scratch on fostering CT. Montiel and Gomez-Zermeño (2021) conducted a systematic review and found 30 articles. They suggested that Scratch is suitable for teaching CT in K–12 education. Although research investigating CT skills in K–12 is prolific, studies investigating how preservice and in-service teachers are prepared for learning and teaching CT skills are relatively scarce, underscoring a need to conduct more empirical research on the teacher population.

2.3 Coding and Computational Thinking in Teacher Education

While the topic of teaching CS in K–12 schools has recently received widespread interest, issues related to teaching coding and CT as part of teacher education have existed for over 40 years (Bull et al., 2020; Schmidt-Crawford et al., 2019). Most notably, the debate in favor of introducing programming to children in K–12 environments stems from Seymour Papert and the publication of *Mindstorms* (Papert, 1980). In his book, Papert argues that by learning computer programming children teach the computer

how to think, which can serve as a catalyst for children to embark on the epistemological journey of thinking about their own thinking. Designed as a tool for learning, Papert and a team of researchers at MIT developed the Logo Programming Language (Logo Foundation, 2014). Early versions of the Logo allowed people to control a robotic turtle, which Papert (1980) described as a "computational object-to-think-with" (p. 11). The turtle eventually migrated to the computer screen as a controllable graphic called a "sprite," which could be used to draw shapes, graphics, and patterns.

In the early 1980s, Logo and other programming languages (e.g., BASIC and Pascal) were starting to find their way into the K–12 classrooms. For example, by January 1983, the state of California had established 15 Teacher Education and Computing Centers with the goal of providing training to teachers in mathematics and CS (Gray, 1983). A few months later, Apple announced their Kids Can't Wait program, which aimed to place 9,250 Apple Iie computers in California elementary and secondary schools (Uston, 1983). Each computer included a copy of the Apple Logo, and representatives from Apple dealers were trained to assist teachers in how to use the programming language.

While Logo had an initial uptake by enthusiastic progressive educators in the US and UK, by the mid-to-late 1980s the majority of teachers dreaded the Logo training sessions out of a fear of being embarrassed in front of their colleagues, or by being "shown up" by students in the classroom who had more expertise at debugging code (Agalianos et al., 2001). Although Logo was initially seen as a promising way to transform curriculum, cognitive and metacognitive studies from the mid-1980s found little to no difference between Logo and non-Logo users (Ames, 2018). Despite these failures in the K–12 setting, researchers at MIT continue to develop new platforms, such as LEGO/logo, which allowed people to build programmable machines with LEGO bricks (Resnick & Ocko, 1990). As part of the LEGO/logo project, a new version of the Logo was created called Logo Blocks (Logo Foundation, 2014). This innovation allowed users to create programs by snapping together jigsaw-like puzzle pieces instead of writing text-based lines of code. This block-based coding innovation was incorporated into a new Logo programming environment called Scratch, which was officially launched to the public in 2007 (Resnick et al., 2009).

While the timing of Wing's 2006 article on CT and the 2007 release of Scratch are not directly correlated, they both serve as a catalyst for the reintroduction of CS into teacher education programs. One of the challenges with introducing these concepts into teacher education is addressing misconceptions about what delineates CS, CT, and coding. As Yadav et al. (2017a) point out, while CS unplugged activities and block-based programming languages like Scratch are an approachable way to introduce preservice and in-service teachers to CT, care must be taken in teacher education programs to ensure that CT is not mistakenly equated with programming or instructional technology. Their survey study, which examined 134 preservice teachers' conceptions of CT and classroom implementation,

found that participants defined CT in terms of problem-solving and logical thinking, and often associated the concept with the use of a computer. They recommend that teacher educators should embed CT within educational technology and content-specific method courses. By doing so, preservice teachers will have more opportunities to think computationally and gain experience with CT as a generic set of skills that do not require a computer.

While CT does not require a computer, robotics and other physical computing tools have been used to introduce preservice and in-service teachers to CT. Jaipal-Jamani and Angeli (2017) studied how 21 preservice teachers learned about CT as part of an elementary science methods course. Their study found that throughout the semester-long course, preservice teachers' interest and self-efficacy toward robotics increased and that participants showed gains in CT skills such as learning how to write algorithms and debug programs. Additionally, Mason and Rich (2019) performed a literature review that synthesized 21 studies on elementary preservice and in-service teachers' attitudes, self-efficacy, or knowledge to teach computing, coding, or computational thinking. As part of their review, six of the studies focused on both CT and robotics. They found that although most interventions were relatively short in duration, training and professional development led to gains in preservice and in-service teachers' computing content knowledge and self-efficacy.

In addition, Bower et al. (2017) have also shown that in-service K–8 teachers can improve their CT pedagogical capabilities through a combination of "unplugged" and block-based coding activities. They conducted a series of CT workshops which found that teachers developed their CT understanding, pedagogical capacities, technological knowledge, and confidence through these targeted professional learning opportunities. While research has shown that teachers can be successful in learning how to code as part of their in-service training, these coding and CT skills do not automatically transfer to their teaching practices (Guven & Kozcu Cakir, 2020). Instead, teachers need to be introduced to CT within the context of the subject area in which they teach (Yadav et al., 2017c).

2.4 The Impact of the COVID-19 Pandemic on Teachers' Professional Learning of CS and CT

The COVID-19 pandemic has also been posing challenges in providing in-service teachers with needed professional learning opportunities on CS and CT. Virtual professional development (PD) programs have become a popular way to solve participation problems. For example, Jocius et al. (2021) transformed their summer PD workshops into a virtual conference format, including emerging technology tools, pre-PD training, synchronous and asynchronous sessions, Snap! Pair programming, live support, and live networking. They found that the digital tools, formats, and support for teacher engagement and collaboration were the most effective changes they made that increased participants' self-efficacy in teaching CT, supporting collaboration, enabling participants to design CT-infused

content-area lessons, and learning about strategies for virtual, hybrid, and face-to-face classroom teaching. Based on the overall success, this group of researchers commented that they plan to continue to develop and use virtual PD.

Similarly, Mouza et al. (2022) decided to utilize a virtual PD institute for K–12 in-service teachers, which includes both synchronous and asynchronous sessions. Participants reported higher scores in knowledge and skills after the virtual PD program, as well as a higher level of confidence and preparation to teach CS in practice. Both Jocius et al. (2021) and Mouza et al. (2022) pointed out the importance of teachers' collaboration and sharing officially and unofficially during virtual PD programs. Jocius et al. (2021) cautioned the researchers to increase the number of facilitators, provide more extensive pre-workshop training, and carefully select virtual tools. Comparably, Mouza et al. (2022) especially recommend diversifying and broadening teacher participation, providing differentiated instruction, increasing hands-on activities, and prioritizing teachers' engagement.

To address the need for content-specific integration of CS and CT and broadened participation, the authors of this study introduced in-service teachers to CT and coding as part of a graduate-level online course. These teachers developed their own content-specific CT lessons and implemented those lessons in their K–12 classrooms, makerspaces, or as part of after-school programs. In particular, this study aims to investigate in-service teachers' perceptions and development of CT skills in this required emerging technologies course as part of an online instructional technology graduate program.

K. **Methods**

In this section, the researchers describe the implementation of a case study methodology to study in-service teachers' perceptions and development of CT skills (Yin, 2017). Using a holistic single-case design, the unit of analysis is bounded to 29 participants who were enrolled in a graduate emerging technologies course during the Fall of 2021.

K.12 *Research Context and Module Design*

Creating with Emerging Technologies is an asynchronous online graduate-level course that is designed to introduce in-service teachers to trends and issues related to instructional technology and design. This course was launched in the Fall of 2021 with four class sections that averaged 20 students per section. The course consists of eight modules, including (1) Introduction to Constructionism, (2) Computational Thinking, (3) Algorithms in Education, (4) Machine Learning and Artificial Intelligence, (5) Learning Spaces (i.e., makerspaces, Fab Labs, and active learning spaces), (6) eXtended Reality (i.e., virtual, augmented, and mixed reality), (7) Open Educational Resources (OER), and (8) The Creative

Classroom. As part of a 15-week course, the first seven modules are designed to take two weeks each, with the last module serving as a one-week final reflection. Each module consists of required reading, online videos, a written reflection, and either a coding, electronics, or 3D 102odelling project. During the first week of each module, students complete the readings, watch the videos, and post a 300-500 word summary as part of a Google Slide design journal. During the second week, students reply to at least two of their peers, and complete a weekly project (e.g., creating a digital story in Scratch). The required materials for the course include the *SparkFun Inventor's Kit for micro:bit*, which includes a micro:bit, breadboard, and various electrical components such as LEDs, resistors, wires, potentiometer, servomotor, and switches (see Figure 1). While the course is designed for the micro:bit V2 (which includes a built-in speaker, microphone, and capacitive touch), this research study used the micro:bit V1 due to supply chain shortages. Kits for the study were purchased with internal grant funds and two of the four class sections were picked via a random number generator to participate in the study.

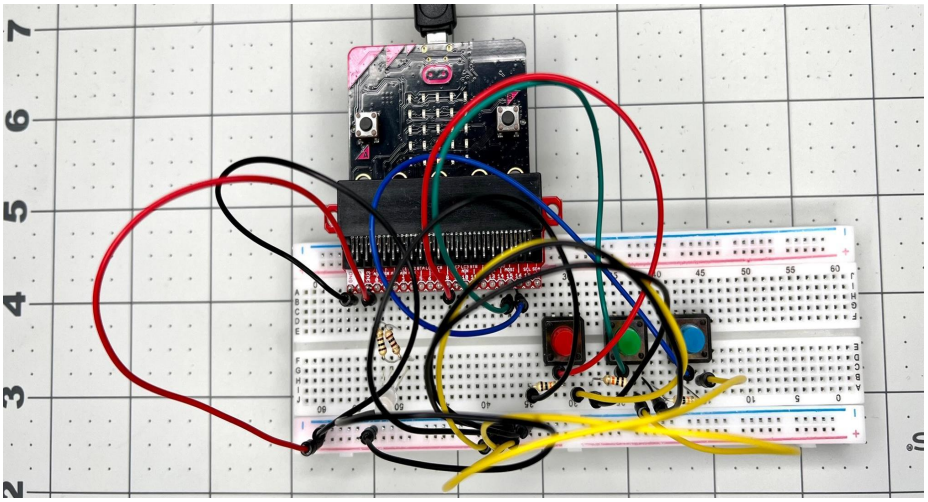


Figure 1. BBC micro:bit with a breadboard, wires, and electronic components.

As part of the course modules, participants are introduced to block-based coding using Scratch (Scratch, n.d.) and Microsoft Makecode for micro:bit (Microsoft Makecode, 2022). Activities with these platforms include creating a digital story in Scratch (Module 1), programming two inputs and outputs with the BBC micro:bit (Module 2), programming and wiring two inputs on outputs with the breadboard (Module 3), and creating an interactive robotic pet (Module 4). These activities are part of the first four modules in the course and are supported by prerecorded video tutorials, plus two weekly synchronous "Hour of Code" sessions for live troubleshooting. Additionally, as part of the second module, students are introduced to CT through required readings (Grover & Pea, 2018; Wing, 2006) and complete an online quiz based on the BBC Bitesize CT learning modules (BBC Bitesize, n.d.). While CT is the focus of the second module, the concepts and terminology are reinforced throughout the entire course. As part of the fifth module, participants developed a lesson proposal for a Creative Computing Project, which

involved teaching CT and a design process (e.g., creative play, design thinking, or engineering design process) in an alternative setting (e.g., a non-traditional classroom, makerspace, or after-school program.) Suggested Creative Computing Projects included hands-on CS Unplugged activities, digital storytelling in Scratch, or breadboarding with Makecode and the BBC micro:bit. After implementing their project, participants wrote a Creative Computing Project final report, which documented the design and implementation of their project and was due by the end of the seventh module. The final report includes a section on CT, where participants are encouraged to use CT terminology as part of their open-ended responses.

K.12 *Participants*

Overall, 29 in-service teachers voluntarily participated in this study. Among them, 24 teachers completed both the pre and post-surveys while one teacher only filled out the presurvey. Four teachers did not respond to the survey requests. Based on the 25 responses to the demographic questions, six teachers identified as men and 19 as women. Five participants were 23-26 years old, two were 27-32 years old, six were 32-40 years old, nine were 40-50 years old, and three were more than 50 years old. Fourteen teachers are white, seven are African Americans, three are Asians, and one is in the other category. Nine participants had Bachelor's degrees while 16 had Master's degrees. The years of teaching experience ranged from 2 to 28 years. These participants also taught in a variety of content areas and some of them taught in several categories: science (8), all subject areas (6), social studies (6), English Language Arts (4), math and science/STEM (3), health and physical education (2), food science and nutrition (1), video production (1), and one participant did not report their content area. Seven teachers worked in elementary schools, ten in middle schools, six in high schools, and two in the K–12 levels.

Twenty-four in-service teachers filled out the survey with questions about their competencies in programming languages. Three teachers said that they had some background in coding such as a Bachelor's degree in Computer Information Systems, coursework in computing languages, and teaching experiences with coding and robotics in their classrooms. However, 21 teachers reported that they did not have any coding background prior to the course. One teacher did not answer the questions. Teachers also reported their competencies with various coding languages (see Table 1). Overall, in-service teachers did not have extensive experience in programming languages. Furthermore, the majority of the teachers never programmed anything. Compared to other programming languages, teachers had relatively more experience in using educational coding languages, such as Scratch and OzoBlockly.

Table 1. In-Service teachers' self-reported competencies in programming languages (n = 24).

Programming languages	Never programmed in this language.	Minimal experience. Maybe compiled a test program.	Some experience. Wrote several small to medium-sized programs.	Substantial experience. Wrote several small to medium-sized programs.	Extensive experience. Wrote many programs.
C++	21	2	/	1	/
JAVA	18	4	2	/	/
Visual Basic	22	1	/	1	/
Python, Perl, or other scripting-based languages	21	3	/	/	/
JavaScript, HTML, ASP, or other web-based languages	17	6	1	/	/
Scratch, OzoBlockly, or another block-based coding	5	11	7	/	1

K.12 *Data Collection and Analysis*

The researchers used a validated survey instrument called the CTS scale to collect data on in-service teachers' perceptions of CT skills. The researcher who designed the survey instrument computed Cronbach's Alpha of the overall scale and reported an internal consistency coefficient of .969 (Yağci, 2019). The survey used in the current study has ten demographic questions and 42 Likert-scale questions on four variables: (a) problem solving (20 questions), (b) collaborative learning & critical thinking (8 questions), (c) creativity (9 questions), and (d) algorithmic thinking (5 questions). A pre and post-survey design was used. An informed consent form was sent to students in the course. Once the participants signed the consent form, a link to the presurvey was sent to them. It took students around 15 minutes to complete the survey. At the end of the coding instructions, a link to the post-survey was sent to the

participants and it took them around 15 minutes to finish the post-survey. Cronbach's Alpha ranges from .45 to .89 (presurvey: .81, .74, .80, .53; post-survey: .89, .62, .79, .45). Cronbach's Alphas of the first three variables indicate they are very reliable, which demonstrates a high level of internal consistency for the scales with this specific sample. Cronbach's Alphas of the last scale, algorithm thinking, show it is a moderately reliable scale with the current sample (Hinton et al., 2004). Pair-sample *t*-tests were used to examine whether there were statistically significant differences in teachers' perceptions of CT.

A test of CT skills was also used In this study. This test has 12 multiple-choice questions and four open-ended questions. Participants took a pretest before learning the modules and afterward, they took the post-test. Paired-sample *t*-tests were conducted to investigate whether there were statistically significant differences in teachers' pre and post-test scores. These test scores are a way of measuring teachers' CT skills, which provides triangulation to the self-reported data on teachers' CT perceptions.

Qualitative data consisted of the participants' Creative Computing Project final report. This report included eight open-ended sections, one of which was devoted to CT. The prompt for the CT section stated, "Using language such as abstraction, decomposition, pattern recognition, and algorithms, describe the computational thinking that you observed as part of your Creative Computing Project. If you could redesign your lesson, what would you do to encourage more computational thinking?" Based on the themes of abstraction, decomposition, pattern recognition, algorithms, and debugging the researchers used deductive coding (Miles et al., 2019) to identify whether the CT terminology was used correctly, incorrectly, or was absent based on the definitions of the BBC Bitesize CT learning modules (BBC Bitesize, n.d.). The researchers calibrated their coding criteria by analyzing two of the participants' CT sections together and then coded the other 27 participants separately. Once coding was complete, the researchers initially agreed on the use of 93% of participants' use of terminology. Based on a Cohen Kappa, interrater reliability (IRR) was found to be 0.86, or a "near-perfect agreement" (Cohen, 1960; Ranganathan et al., 2017). The data was then reanalyzed to resolve any disagreements until 100% IRR was achieved.

K. **Results**

The researchers analyzed both quantitative and qualitative data to answer the three research questions, focusing on in-service teachers' perceptions and development of CT skills. Findings were triangulated using three types of data from the self-reported survey, CT pre and post-test, and the CT section of participants' final written report on their CT implementation. In the following section, results are written to answer each research question.

4.1. RQ 1: What were in-service teachers' perceptions about their CT skills before and after taking the graduate emerging technologies course?

In-service teachers' CT perceptions changed after taking the modules on coding and creative computing (see Table 2). There was a statistically significant improvement in their perceptions of problem-solving, $t(24) = -3.99$, $p < .001$, from 80.16 ± 6.81 to 86.44 ± 7.43, an improvement of 6.28 ± 7.88. A statistically significant decrease was found in teachers' perceptions of collaborative learning and critical thinking, $t(24) = 1.99$, $p = .03$, from 19.16 ± 5.23 to 17.36 ± 4.12, a decrease of 1.80 ± 4.52. Last, the researchers discovered a statistically significant increase in teachers' perceptions of creativity, $t(24) = -2.21$, $p = .02$, from 35.28 ± 4.69 to 36.92 ± 3.82, an increase of 1.64 ± 3.71. Changes in problem-solving had a large effect size of .88, while differences in collaborative learning & critical thinking and creativity had small effect sizes of .38. Algorithmic thinking had no statistically significant change.

Table 2. Results from the paired sample *t*-tests on in-service teachers' CT perceptions (*n*=25).

CT perceptions	Pre		Post		Paired sample *t*-tests		
	M	*SD*	*M*	*SD*	*t*	*p*	Cohen's d
Problem solving	80.16	6.81	86.44	7.43	-3.99	<.001***	.88
Collaborative learning & critical thinking	19.16	5.23	17.36	4.12	1.99	.03*	.38
Creativity	35.28	4.69	36.92	3.82	-2.21	.02*	.38
Algorithmic thinking	19.28	2.19	18.72	2.11	1.22	.12	.26

Note. * $p < .05$; ** $p < .01$; *** $p < .001$.

4.2 RQ 2: Was there a difference in in-service teachers' CT test scores after taking the course?

In-service teachers took the same test focusing on CT skills before and after the coding and creative computing modules. The test has a total of 100 points. Their pre and post-test scores of CT skills had a wide range, with pre-scores ranging from 28 to 100 and post-scores ranging from 25 to 100. Their pre

and post-test scores changed after taking the coding and creative computing modules. There was a statistically significant improvement in their CT scores, $t(23) = -1.74$, $p < .05$, from 65.17 ± 19.04 to 73.04 ± 18.52, an improvement of 7.88 ± 22.18. The effect size is .42, a medium effect size.

The researchers conducted Ir paired sample *t*-test to further examine the difference in the test scores of the 12 multiple-choice questions. There was a statistically significant improvement in their scores on the multiple-choice questions, $t(23) = -3.57$, $p < .001$, from 36.88 ± 11.96 to 45.63 ± 10.35, an improvement of 8.75 ± 12.00. The effect size is .78, a large effect size. Overall, according to the CT test scores, in-service teachers developed their CT skills after studying the modules.

4.3 RQ 3: How frequently and accurately did in-service teachers apply CT terminology in their final reports?

As described in the Data Analysis section, two researchers coded the qualitative data focusing on the frequency and accuracy of the CT concepts, which were collected from participants' final reports after implementing their course projects. Table 3 illustrates a few examples of how in-service teachers wrote about the terminology of CT skills.

Table 3. Examples of teachers' writing on the terminology of CT skills.

CT terminology	**Examples from qualitative data**	
	Used correctly	**Used incorrectly**
Abstraction	An example of pattern recognition used by the students is knowing that an animal classified as a mammal has to give live birth, have warm blood, have fur or hair, and breathe with lungs. Students used the process of **abstraction** to be able to filter out any unnecessary information that is not needed in order to introduce their newly discovered animal.	***Abstraction:*** The students reread the ending and we decided to ignore the entirety of Chapter 23 which is the last chapter of the novel. The students had lots of debate about whether or not the project should start from the moment Jonas leaves versus the last chapter. To help the students, we watched the last ten minutes of "The Giver" movie which really appealed to all the students. Due to some PG-13 thematic elements, I could not

		show the entire movie.
Algorithms	To develop solutions to solving this problem, the students will use **algorithmic thinking**. To gain an understanding of this process, I will ask the students to make a sandwich. In doing this, we will discuss the sequence and order of making a sandwich using **algorithmic thinking**. In using the Scratch program, code blocks are called scripts. A script is an ordered list of instructions that can also be called an **algorithm**. The character in the program is called a sprite. The stage refers to the background of the story or the game.	***Algorithms:*** Students used the tutorials for adding saved images as sprites and backdrops in Scratch.
Decompositi on	This was followed by having students give verbal directions in pairs to accomplish a simple task such as writing "hello" with a pen. This introduces students to some of the concepts of computational thinking by asking students to engage in **decomposition** and breaking the task down into smaller parts.	When coding using cups as a hands-on manipulative, scholars were able to recognize patterns to create the codes and **decomposition** to solve premade codes.
Pattern recognition	Teacher reviewed **patterns** in strings of shapes to remind students of the concept of **patterns**. The teacher explained to students that **pattern recognition** can make coding easier. The teacher asked students to open their Scratch codes to look for **patterns**. The teacher explained to students how to use code to make their Sprites repeat actions. Students demonstrated using Scratch code the concept of repeating an action in their digital storyboard.	The students will use **pattern recognition** to help with coding the movements and speech for each background to help make the coding more organized and appropriate for each scene.
Debugging	To test their thinking, students had opportunities to try out the command language created by other groups – they worked collaboratively to **debug** any steps and provided feedback to their peers for ways	/

	to make the process more efficient for other users.	

Table 4 shows the numbers and percentages of the terms that were used correctly, incorrectly, or not mentioned at all. It is noticeable that most teachers used two CT terms correctly, algorithms and decomposition. However, only 59% of teachers used the terms abstraction and pattern recognition correctly. Furthermore, most teachers did not mention debugging at all, possibly due to the term being absent from the final report's question prompt. The finding highlights the need to emphasize certain CT terms, specifically abstraction, pattern recognition, and debugging, in future iterations.

Table 4. Usage of the CT terminology in teachers' final reports (*n* = 29).

CT terminology	**Used correctly**		**Used incorrectly**		**Absent**	
	n	**%**	**n**	**%**	**n**	**%**
Abstraction	17	59%	4	14%	8	28%
Algorithms	25	86%	1	3%	3	10%
Decomposition	23	79%	3	10%	3	10%
Pattern recognition	17	59%	3	10%	9	31%
Debugging	4	14%	/	0%	25	86%

Additionally, the researchers ran multiple Pearson's correlation tests using the demographic variables and the data from the survey, test, and final reports. However, no statistically significant correlation was found. This finding revealed that no relationships were found between the demographic variables, survey results, tests, and usage scores. Moreover, it means that the self-reported data from the CT perceptions survey did not correlate with the performance-based data from the CT test and terminology usage scores.

5. Discussion

5.1 Impact on In-service Teachers' CT Perceptions

The purpose of this study was to investigate in-service teachers' perceptions and development of CT skills in an online graduate emerging technologies course. Data analysis indicated that participants reported that they developed some aspects of their CT skills, such as problem-solving and creativity. Moreover, the change in their perceptions of problem-solving had a large effect size. These findings

demonstrated that the online course had a positive impact on teachers' perceptions of CT skills, especially problem-solving and creativity. These results were also motivating since the course modules were designed to focus on creative computing with ample opportunities for problem-solving. Similar findings were found in other virtual PD programs (Jocius et al., 2021; Mouza et al., 2022).

Nevertheless, at the same time, teachers' perceptions of collaborative learning and critical thinking skills decreased after taking the course. One plausible reason might be the lack of peer coding opportunities. The authors recognized the benefits of peer coding as evidenced by findings in the field (Campe et al., 2020; Hanks et al., 2011). Even so, since this course was an online course, it was challenging to design peer coding activities that allowed multiple in-service teachers to program the same project due to various reasons such as lack of time and lack of proper Web 2.0 tools for peer coding. Jocius et al. (2021) used Snap! Pair programming and live support methods in their virtual PD program, which might be promising strategies to use. The authors also plan to explore live peer coding tools like Glitch.com and Twitch.tv for future iterations. Furthermore, this finding warrants more research on peer coding in online courses and the effectiveness of various tools and approaches for peer coding activities in various learning modalities.

While the effect size is small, there is evidence that these creative computing activities have the potential for fostering more creativity in the classroom. All computational projects in the course were designed to be open-ended with inclusive pedagogies in mind, to ensure that all participants could be creative in how they express their ideas and identities. Creative computing is an emerging branch of computer science that is gaining recognition through the integration of coding, interactive art, and making (Blikstein, 2018). This approach is less used in research and practice, but deserves more attention for it involves aesthetic experience, design-based learning, and storytelling (Hsu et al., 2018). The computational tools and devices used in this study are just one feasible way of enabling teachers to engage in creative computing while also making connections between CT and their subject areas. The authors recognize that there are other creative computing curricula that are publicly available (Creative Computing Lab, n.d.) and encourage teachers and teacher educators to explore how CT can be used to foster creativity in the classroom.

5.2 Impact on In-service Teachers' Development of CT Skills

Besides examining in-service teachers' perceptions of CT skills, the authors also analyzed the pre and post-test scores on CT skills. Findings revealed that overall in-service teachers improved their test scores after the modules, which demonstrated the development of CT skills. These results infer that the modules are effective in developing in-service teachers' CT skills. Several design factors might contribute to the modules' effectiveness. First, the course content was chunked to build on knowledge from previous

modules. In-service teachers used Scratch, a block coding programming language, to create their digital storytelling projects first. Once they developed foundational CT and coding skills using block-based coding, they wrote codes on the Microsoft Makecode platform to program their BBC micro:bit. Last, they transitioned to breadboarding and creating their robotic pet, which was more challenging due to the need to troubleshoot both the digital code and the physical electrical components. To summarize, the projects were purposefully designed to follow an easy-to-difficult progression in order to achieve maximal improvement (Wisniewski et al., 2019).

Another design feature is the synchronous "Hour of Code" office hours, which were offered twice a week for in-service teachers to create, discuss code, and hang out with the course instructor. Although these sessions were optional, in-service teachers joined the sessions from time to time. Moreover, these sessions were recorded for in-service teachers to watch anytime anywhere. This method offered in-service teachers more instructional time and opportunities to ask questions, create, and troubleshoot in a synchronous group setting. Providing live support and prioritizing teachers' engagement have been justified as useful strategies for virtual professional learning in the literature (Jocius et al., 2021; Mouza et al., 2022).

A third design feature Is the open-ended course projects, which utilized a "low threshold, high ceiling" approach. This strategy allows in-service teachers to engage in a variety of projects and provides room for them to consider their contexts and subject areas. To facilitate this method, the course instructor curated and created ample course materials that matched teachers' different abilities and learning preferences. Future research should examine the design features of such a course, propose instructional models, and design criteria to help teacher educators better design such courses.

Nonetheless, results from the descriptive data revealed that there was a big gap in the testing scores of these in-service teachers. Some teachers earned full marks on the pre and/or post-tests while other teachers scored relatively low for both tests. This result is somewhat alarming because it shows that some in-service teachers are not well-equipped with enough CT skills and it will be challenging for them to design CT-related curricula. It also indicates that more preparation on the knowledge and application of CT is needed.

Pedagogical approaches that might be helpful to facilitate further preparation or professional development efforts are adaptive learning (Hooshyar et al., 2021), personalized learning (Moon et al., 2020), and instructional technology coaching (Garvin et al., 2019; Israel et al., 2015). The authors recommend teacher educators pay attention to the gap in teachers' prior knowledge of CT and coding and design preparation and professional development accordingly.

5.3 Correlations between Self-reported and Performance-based Data

Pearson's Correlation tests revealed no statistically significant correlations between the demographic variables, self-reported data, and performance-based data. In other words, in-service teachers' perceptions of their CT skills did not correlate with their actual CT skills demonstrated in the performance-based data. Furthermore, there was no correlation between the two types of performance-based data, the CT test scores and the CT terminology scores. These findings have direct implications for future research, which could explore the correlation between other self-reported data, such as CT attitudes and self-efficacy, and various types of performance-based data measuring CT skills and CT implementation. In addition, more validated and standardized instruments are needed to measure teachers' CT implementation.

5.4 Beyond the Four Cornerstones of Computational Thinking

As demonstrated by the findings of the qualitative data, terminology related to the four cornerstones of computational thinking (i.e., abstraction, decomposition, pattern recognition, and algorithms) were used by the majority of participants. While these cornerstones were established early in the development of CT frameworks, the concepts related to CT skills and practices have expanded to include numerous other concepts such as debugging, selecting tools, automation, computational 112odelling, and data practices (Mills et al., 2021). As teacher educators expand the learning of CT in teacher preparation and professional development programs, it is crucial to look beyond the four cornerstones to ensure teachers and students receive a solid foundation in the concepts and practices that will prepare them for later engagement in CS. For example, professionals in CS engage in an iterative process of testing, debugging, and evaluating to ensure their programs function as designed. Similar to learning how to play a musical instrument, both CT and CS require practice and repetition in order to improve skills, develop fluency, and accomplish larger goals.

The authors recommend that those developing professional development and courses related to CT should investigate frameworks that move beyond the four cornerstones and include a broader range of CT skills and practices (e.g., Grover & Pea, 2018; Mills et al., 2021). While the four cornerstones initially serve as a good introduction to short-term professional development, the concepts associated with CT have widely expanded over the past 15 years. Additionally, more emphasis should be placed on developing a conceptual understanding of abstraction, which Jeanette Wing (2010) considers to be the most high-level thought process in CT. Teacher educators should provide ongoing professional development that seeks to cultivate a deeper understanding of CT and CS concepts with the goal of achieving a higher degree of K–12 integration.

6. Limitations

Limitations of this study include a relatively small sample size of 29 participants, of which 24 completed both surveys. Despite the small sample, researchers were able to produce meaningful results from the data across various statistical tests. Another limiting factor includes the use of a self-reported survey instrument to measure in-service teachers' CT perceptions before and after taking the course. All participants were enrolled in an emerging technology course as part of an Instructional Technology graduate program. As a result, participants likely identified as advocates for technology in the classroom and may have more experience with CT than teachers enrolled in other graduate programs. While CT was included as the focus of the second module, the concepts and terminology are reinforced throughout the entire course. This includes a CT section in the final written Creative Computing Project report. This study design focused on the change in CT perceptions and skills before and after the course, further studies are needed to measure the impact of individual modules or topics. Furthermore, this study took place as part of an asynchronous online course, thus findings may not be generalizable to synchronous, in-person, or hybrid settings.

7. Conclusion

This study found that in-service teachers enrolled in an online asynchronous graduate emerging technologies course were able to improve their CT problem-solving and creativity skills through a series of learning modules and activities with large effect sizes, which indicates the effectiveness of a virtual course. Despite these gains, participants reported a decrease in their collaborative learning and critical thinking skills, however, with a small effect size. Most teachers were able to correctly apply the terms algorithms and decomposition in their final reports. However, only 59% of teachers correctly used the term abstraction and pattern recognition, and most teachers did not mention debugging at all.

In general, more needs to be done to help in-service teachers develop their CT skills. As this study has demonstrated, it is possible for in-service teachers to develop these skills asynchronously and online with a certain degree of success. However, more research is needed to better understand how to facilitate the development of CT collaborative learning and critical thinking skills in different teaching and learning formats, such as face-to-face, hybrid, and especially virtual. Those teaching CT skills should model and practice the correct use of terminologies, such as abstraction and pattern recognition, which were the most frequently misused terms in this study. In addition, greater emphasis should be placed on testing and debugging in order to move beyond the four cornerstones of CT. More empirical research is needed that addresses how in-service teachers develop and implement their CT skills. In addition, course developers should engage in design-based research to help the academic community better understand

how teachers can develop a deeper understanding of CT, implement CT skills in their subject areas, and cultivate a sustained interest in CS.

References

Agalianos, A., Noss, R., & Whitty, G. (2001). Logo in mainstream schools: The struggle over the soul of an educational innovation. *British Journal of Sociology of Education, 22*(4), 479–500. https://doi.org/10.1080/01425690120094449

Ames, M. G. (2018). Hackers, computers, and cooperation: A critical history of Logo and constructionist learning. In *Proceedings of the ACM on Human-Computer Interaction, 2*(18), 1–19. https://doi.org/10.1145/3274287

Barcelos, T. S., Muñoz-Soto, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics learning through computational thinking activities: A systematic literature review. *Journal of Universal Computer Science*, 24(7), 815–845.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

BBC Bitesize. (n.d.). Introduction to computational thinking. https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field* [Report]. Google LLC. https://goo.gl/gmS1Vm

Bower, M., Wood, L. N., Lai, J. W., Highfield, K., Veal, J., Howe, C., Lister, R., & Mason, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education, 42*(3), 53–72. https://doi.org/10.14221/ajte.2017v42n3.4

Bull, G., Garofalo, J., & Hguyen, N. R. (2020). Thinking about computational thinking: Origins of computational thinking in educational computing. *Journal of Digital Learning in Teacher Education, 36*(1), 6–18. https://doi.org/10.1080/21532974.2019.1694381

Brennan, K. (2015). Beyond technocentrism. *Constructivist Foundations, 10*(3), 289–296. https://constructivist.info/10/3/289.brennan

Campe, S., Denner, J., Green, E., & Torres, D. (2020). Pair programming in middle school: variations in interactions and behaviors. *Computer Science Education, 30*(1), 22–46. https://doi.org/10.1080/08993408.2019.1648119

Cansu, S. K., & Cansu, F. K. (2019). An overview of computational thinking. *International Journal of Computer Science Education in Schools, 3*(1), 1–11. https://doi.org/10.21585/ijcses.v3i1.53

Code.org, CSTA, & ECEP Alliance. (2022). *2022 State of computer science education: Accelerating action through advocacy*. https://advocacy.code.org/stateofcs

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*, 37–46. https://doi.org/10.1177/001316446002000104

Creative Computing Lab. (n.d.). *Creative computing curriculum*. Harvard Graduate School of Education. https://creativecomputing.gse.harvard.edu/guide/

CSTA (2017). *K–12 Computer science standards*. Retrieved from https://drive.google.com/file/d/1-dPTAI1yk2HYPKUWZ6DqaM6aVUDa9iby/view

Dufva, T., & Dufva, M. (2016). Metaphors of code—structuring and broadening the discussion on teaching children to code. *Thinking Skills and Creativity, 22*, 97–110. https://doi.org/10.1016/j.tsc.2016.09.004

Garvin, M., Killen, H., Plane, J., & Weintrop, D. (2019, February). Primary school teachers' conceptions of computational thinking. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 899–905). https://doi.org/10.1145/3287324.3287376

Goode, J. (2008, March). Increasing diversity in K–12 computer science: Strategies from the field. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (pp. 362–366). https://doi.org/10.1145/1352322.1352259

Gretter, S., Yadav, A., Sands, P., & Hambrusch, S. (2019). Equitable learning environments in K–12 computing: Teachers' views on barriers to diversity. *ACM Transactions on Computing Education (TOCE), 19*(3), 1–16. https://doi.org/10.1145/3282939

Gray, L. E. (1983). TECC/8: A Teacher Education and Computing Center. *Teacher Education Quarterly, 10*(4). 8–21.

Grover, S., & Pea, R. (2013). Computational thinking in K–12. A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.) *Computer science education: Perspectives on teaching and learning in school* (pp. 19–38). Bloomsbury.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Guven, G., & Kozcu Cakir, N. (2020). Investigation of the opinions of teachers who received in-service training for Arduino-assisted robotic coding applications. *Educational Policy Analysis and Strategic Research, 15*(1), 253–274. https://doi.org/10.29329/epasr.2020.236.14

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education, 21*(2), 135–173. https://doi.org/10.1080/08993408.2011.579808

Hello Ruby. (2019, September 2). *Episode 02: computational thinking* [Video]. YouTube. https://www.youtube.com/watch?v=K3vwRQCfTHc

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education, 4*(1), 48–69. https://doi.org/10.1007/s40751-017-0038-8

Hinton, P. R., Brownlow, C., McMurray, I., & Cozens, B. (2004). *SPSS Explained*. Routledge Inc. East Sussex, England.

Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G. J., Wang, M., Lim, H., & Delev, D. (2021). From gaming to computational thinking: An adaptive educational computer game-based learning approach. *Journal of Educational Computing Research, 59*(3), 383–409. https://doi.org/10.1177/0735633120965919

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Huang, W., & Looi, C. K. (2021). A critical review of literature on "unplugged" pedagogies in K–12 computer science and computational thinking education. *Computer Science Education, 31*(1), 83–111. https://doi.org/10.1080/08993408.2020.1789411

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education, 82,* 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

ISTE (2016a). *ISTE standards for educators*. https://www.iste.org/standards/for-educators

ISTE (2016b). *ISTE standards for students*. https://www.iste.org/standards/iste-standards-for-students

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology, 26*(2), 175–192. https://doi.org/10.1007/s10956-016-9663-z

Jocius, R., Joshi, D., Albert, J., Barnes, T., Robinson, R., Cateté, V., Dong, Y., Blanton, M., O'Byrne, I., & Andrews, A. (2021, March). The virtual pivot: Transitioning computational thinking PD for middle and high school content area teachers. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 1198–1204). https://doi.org/10.1145/3408877.3432558

K–12 Computer Science Framework. (2016). https://k12cs.org

Kalogiannakis, M., Tzagkaraki, E., & Papadakis, St. (2021, March 18-19). A systematic review of the use of BBC micro:bit in primary school. In *Proceedings of the 10th Virtual Edition of the International Conference New Perspectives in Science Education*, 379–384, Florence, Italy. https://doi.org/10.26352/F318_2384-9509

Kennedy, C., Kraemer, E. T., & Benson, L. C. (2021). Active learning techniques for computing education. In C. Mouza, A. Yadav, & A. Ottenbreit-Leftwich (Eds.) *Preparing pre-service teachers to teach computer science: Models, practices, and policies* (pp. 3–28). Information Age Publishing, Inc.

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of Science Education and Technology, 29*(1), 174–188. https://doi.org/10.1007/s10956-019-09798-4

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education, 3*(2), 154–171. https://doi.org/10.1007/s40751-017-0031-2

Logo Foundation. (2014). *Logo history*. https://el.media.mit.edu/logo-foundation/what_is_logo/history.html

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education, 19*(4), 790–824. https://citejournal.org/volume-19/issue-4-19/general/preparing-elementary-school-teachers-to-teach-computing-coding-and-computational-thinking

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239–264. https://doi.org/10.1080/08993408.2013.832022

Microsoft Makecode. (2022) *Microsoft Makecode for micro:bit* (Version 4.0.18) [Computer software]. Microsoft. https://makecode.microbit.org/

Miles, M. B., Humberman, A. M., & Saldaña, J. (2019). *Qualitative data analysis: A methods sourcebook* (4th ed.). Sage Publishing.

Mills, K., Coenraad, M., Ruiz, P., Burke, Q., & Weisgrau, J. (2021, December). *Computational thinking for an inclusive world: A resource for educators to learn and lead*. Digital Promise. https://doi.org/20.500.12265/138

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record, 108*(6), 1017–1054. https://doi.org/10.1111/j.1467-9620.2006.00684.x

Montiel, H., & Gomez-Zermeño, M. G. (2021). Educational challenges for computational thinking in K–12 education: A systematic literature review of "Scratch" as an innovative programming tool. *Computers, 10*(6), 69. https://doi.org/10.3390/computers10060069

Moon, J., Do, J., Lee, D., & Choi, G. W. (2020). A conceptual framework for teaching computational thinking in personalized OERs. *Smart Learning Environments, 7*(1), 1–19. https://doi.org/10.1186/s40561-019-0108-z

Mouza, C., Mead, H., Alkhateeb, B., & Pollock, L. (2022). A Virtual Professional Development Program for Computer Science Education During COVID-19. *TechTrends*, *66*(3), 436–449. https://doi.org/10.1007/s11528-022-00731-y

NGSS Lead States (2013). Next generation science standards: For states, by states. The National Academies Press, Washington, DC. https://nap.nationalacademies.org/catalog/18290/next-generation-science-standards-for-states-by-states

Orton, K., Weintrop, D., Beheshti, E., Horn, M., Jona, K., & Wilensky, U. (2016, July). Bringing computational thinking into high school mathematics and science classrooms. In C. K. Looi, J. L. Polman, U. Cress & P. Reimann (Eds.), *Transforming Learning, Empowering Learners: The International Conference of the Learning Sciences (ICLS) 2016* (pp. 705–712). Singapore: International Society of the Learning Sciences. https://repository.isls.org/handle/1/183

Papadakis, S. (2021). The impact of coding apps on young children Computational Thinking and coding

skills. A literature review. *Frontiers in Education, 6*, 657895. https://doi.org/10.3389/feduc.2021.657895

Papadakis, S. (2022). Can preschoolers learn computational thinking and coding skills with ScratchJr? A systematic literature review. *International Journal of Educational Reform*, 1–34. https://doi.org/10.1177/10567879221076077

Papadakis, S., Vaiopoulou, J., Sifaki, E., Kalogiannakis, M., & Stamovlasis, D. (2021). Attitudes towards the use of educational robotics: Exploring pre-service and in-service early childhood teacher profiles. *Education Sciences, 11*(5), 204. https://doi.org/10.3390/educsci11050204

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168. https://doi.org/10.1016/0732-118X(84)90018-7

Ranganathan, P., Pramesh, C. S., & Aggarwal, R. (2017). Common pitfalls in statistical analysis: Measures of agreement. *Perspectives in Clinical Research, 8*(4), 187–191. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5654219/

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Resnick, M., & Ocko, S. (1990). *LEGO/logo—learning through and about design*. Cambridge: Epistemology and Learning Group, MIT Media Laboratory.

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K–12: In-service teacher perceptions of computational thinking. In M. S. Khine (Ed.), *Computational thinking in the STEM disciplines* (pp. 151–164). Springer, Cham. https://doi.org/10.1007/978-3-319-93566-9_8

Schmidt-Crawford, D. A., Lindstrom, D. & Thompson, A. D. (2018). Coding for teacher education: A recurring theme that requires our attention. *Journal of Digital Learning in Teacher Education, 34*(4), 198–200. https://doi.org/10.1080/21532974.2018.1499992

Scratch. (n.d.). *Scratch* (Version 3.0) [Computer software]. Scratch Foundation. https://scratch.mit.edu/

Solomon, C. (1988). *Computer environments for children: A reflection on theories of learning and education*. MIT Press.

Sun, L., Hu, L., & Zhou, D. (2021). Which way of design programming activities is more effective to promote K-12 students' computational thinking skills? A meta-analysis. *Journal of Computer Assisted Learning, 37*(4), 1048–1062. https://doi.org/10.1111/jcal.12545

Uston, K. (1983, October). 9,250 Apples for the teacher. *Creative Computing, 9*(10), 178–183. https://www.atarimagazines.com/creative/v9n10/178_9250_Apples_for_the_teac.php

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D. (2021). The role of block-based programming in computer science education. In *Understanding computing education (Vol 1)*. Proceedings of the Raspberry Pi Foundation Research Seminar series. https://rpf.io/seminar-proceedings-2020

Wilkerson, M. H., & Fenwick, M. (2017). Using mathematics and computational thinking. In C. V. Schwarz, C. Passmore, & B. J. Reiser (Eds.), *Helping students make sense of the world using next generation science and engineering practices* (pp. 181–204). Arlington, VA: National Science Teachers' Association Press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2010). *Computational thinking: What and why?* [Unpublished manuscript]. Computer Science Department, Carnegie Mellon University. https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wisniewski, M. G., Church, B. A., Mercado, E., Radell, M. L., & Zakrzewski, A. C. (2019). Easy-to-hard effects in perceptual learning depend upon the degree to which initial trials are "easy." *Psychonomic Bulletin & Review, 26*(6), 1889–1895. https://doi.org/10.3758/s13423-019-01627-4

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017a). Computational thinking in teacher education. In P. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205–220). Springer, Cham. https://doi.org/10.1007/978-3-319-52691-1_13

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017b). Computational thinking as an emerging competence domain. In M. Mulder (Ed.), *Competence-based vocational and professional education* (pp. 1051–1067). Cham: Springer.

Yadav, A., Stephenson, C., & Hong, H. (2017c). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011, March). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 465–470). https://doi.org/10.1145/1953163.1953297

Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies, 24*(1), 929–951. https://doi.org/10.1007/s10639-018-9801-8

Yaşar, O., Maliekal, J., Veronesi, P., Little, L., & Vattana, S. (2015, March). Computational pedagogical content knowledge (CPACK): integrating 121odelling and simulation technology into STEM teacher education. In *Society for Information Technology & Teacher Education International Conference* (pp. 3514–3521). Association for the Advancement of Computing in Education (AACE). https://www.learntechlib.org/primary/p/150489/

Yin, R. K. (2017). *Case study research and application: Design and methods* (6th ed.). Sage Publishing.

Yurkofsky, M. M., Blum-Smith, S., & Brennan, K. (2019). Expanding outcomes: Exploring varied conceptions of teacher learning in an online professional development experience. *Teaching and Teacher Education, 82*, 1–13. https://doi.org/10.1016/j.tate.2019.03.002

Do Stereotypical vs. Counter-stereotypical Role Models Affect Teacher Candidates' Stereotypes and Attitudes toward Teaching Computer Science?

Lucas Vasconcelos[1]

Fatih Ari[1]

Ismahan Arslan-Ari[1]

Lily Lamb[1]

[1]University of South Carolina

DOI: https://doi.org/10.21585/ijcses.v6i2.174

Abstract

Computer Science (CS) stereotypes promote the mindset that nerdy White males who have a high IQ and are technology enthusiasts are the ones to succeed in the field, leading to gender and racial disparities. This quasi-experimental study investigated if exposing teacher candidates to a stereotypical vs. counter-stereotypical CS role model affects their stereotypes and attitudes toward teaching CS. Participants exposed to a counter-stereotypical role model reported a statistically significant decrease in stereotypes about social skills, and slightly weaker stereotypes about appearance, cognitive skills, and work preferences. Participants exposed to a stereotypical role model reported no changes in stereotypes. Participants in both groups showed increasingly positive attitudes toward teaching CS. Implications for CS teacher education are discussed.

Keywords: stereotypes, role models, computer science, teacher candidates, attitudes

1. Introduction

Computer Science (CS) is a field known for gender and racial disparities (Berg et al., 2018; Cheryan et al., 2015). Mostly White males are enrolled in CS higher education degrees (National Science Foundation, 2019) and make up the computing industry (Bureau of Labor Statistics, U.S. Department of Labor, 2021). A factor undermining participation of females and racial minorities is stereotypes, which promote the mindset that a nerdy White male who has a high IQ and is a technology enthusiast (Master et al., 2016; Pantic et al., 2018) will likely succeed in the field. These stereotypes can affect those who feel dissimilar by undermining their attitudes toward pursuing a CS degree or profession. Among female teacher candidates, stereotypes may promote negative attitudes toward teaching CS, and in turn can undermine integration of CS into the K-12 curriculum. It is important to examine

teacher candidates' stereotypes and attitudes toward teaching CS so these can be addressed and challenged within teacher education programs. This might be helpful for teacher educators striving to prepare teacher candidates who can infuse CS into their future teaching in inclusive and equitable ways.

2. Related Literature

2.1 Stereotypes in CS

A stereotype is a standardized representation created to distinguish a group of individuals based on one or more specific characteristics (Kanahara, 2006; Sills, 1968; Taylor et al., 1994). In CS, stereotypes set apart individuals who are representative of the field, and therefore are considered to become successful professionals in the field. A computer scientist is stereotypically depicted as a White male who looks nerdy (e.g., wear glasses and tooth tracks), has limited social skills, prefers working with machines rather than people, possesses a high level of intelligence and IQ, and is passionate about technologies which results in countless hours working in front of a computer or with other computing devices (Ari et al, 2022; Cheryan et al., 2009, 2015; Cheryan, Meltzoff, et al., 2011; Cheryan, Plaut, et al., 2013; Pantic et al., 2018; Varma, 2020; Vasconcelos et al., 2022).

CS stereotypes can be biased and discriminatory because those who feel dissimilar from the stereotypical computer scientist may end up feeling at the margin. Particularly, females and other minorities may struggle to envision themselves as a CS professional (Cheryan et al., 2009; Cheryan, Meltzoff, et al., 2011; Master et al., 2016; Pantic et al., 2018), which then curtails their aspirations to pursue further education and jobs in the computing industry (Olsson & Martiny, 2018; Shapiro & Williams, 2012). Underrepresentation in CS raises issues about racial justice and socioeconomic equity because those minorities are unable to take on high-paying jobs in the computing industry (Beyer, 2014; Olsson & Martiny, 2018). At the personal level, this undercuts their income potential and limits quality of life. At the societal level, a CS pipeline that is neither inclusive nor diverse misses out on the creativity and innovativeness that come with promoting diversity of perspectives (Cheryan et al., 2015). Central to broadening the CS pipeline is identifying and debunking stereotypes to prevent minorities from feeling unwelcome in the field (Cheryan, Siy, et al., 2011).

2.2 Social Role Theory and Stereotypes

Social role theory posits that behavior is dependent upon the allocation of social roles for males and females within a society (Eagly et al., 2000; Wood & Eagly, 2012). Gender roles are formed through social interactions (Good et al., 2010), which in turn guides the behavior of men and women toward pursuing a certain type of labor (Eagly & Karau, 2002). For instance, men are predominantly assigned

to full-time, paid leadership positions compared to women, who are more often expected to take on caregiving jobs.

Stereotypes are disseminated through the media (Cheryan, Drury, et al., 2012; Cheryan, Plaut, et al., 2013; Graham & Latulipe, 2003), in textbooks (Papadakis, 2018), and in interactions with other members in the community (Good et al., 2010). Using social role theory as a lens, we understand that social interactions that challenge, discredit, and provide alternative representations are critical to identify, debunk, and prevent stereotype formation. One way to achieve this is through social interactions with alternative and diverse role models.

2.3 CS Role Models

Exposure to alternative representations that discredit the default stereotypical representation of a computer scientist is critical to promote interest in CS among minorities. A study with undergraduate students found that females who briefly encountered and talked to a person representing a computer scientist and embodying counter-stereotypical traits (e.g., sports player, music listener, fan of American Beauty movie) displayed higher interest in a CS college major and felt a higher sense of belonging to the field compared to their counterparts exposed to a role model with stereotypical traits (e.g., video game player, programmer, fan of Star Wars movie) (Cheryan, Drury, et al., 2012). Another study conducted two similar experiments by exposing undergraduate students to a STEM stereotypical or counter-stereotypical role model. One experiment was in a face-to-face environment, and one in a virtual environment. Findings from both experiments showed that females in the counter-stereotypical group felt more similar to the role model and anticipated higher success in CS compared to their peers in the stereotypical group (Cheryan, Siy, et al., 2011).

A study with high school students enrolled in engineering classes in schools across the U.S. revealed that being taught by a female faculty over a year resulted in weaker stereotypes among boys who had reported strong stereotypical beliefs about STEM at the beginning of a year (Riegle-Crumb et al., 2017). The same study found that boys who had initially reported weak stereotypical beliefs experienced a decrease in stereotypes when exposed to a high number of female peers in the classroom (Riegle-Crumb et al., 2017). A counter-stereotypical role model also influences young girls. In Buckley et al.'s (2021) study, short stories about female scientists, which represented counter-stereotypical characters, were read to young girls aged 6-8 years old. Findings revealed that young girls who listened to those stories were more likely to recognize females as very smart individuals over males compared to other girls who were not exposed to those stories. Similarly, Gilbert (2015) found that asking women to reflect and write about biographies of female role models led to weaker STEM

stereotypes and stronger associations between women and science, as well as increased sense of belonging in STEM compared to their peers not exposed to a female role model.

Previous research shows the impact of counter-stereotypical role models, but most studies have been conducted with secondary or college students. By the time this paper was submitted, no study had investigated the impact of stereotypical vs. counter-stereotypical CS role models on teacher candidates. Grounded on social role theory, we hypothesize that exposing female teacher candidates to a counter-stereotypical role model results in weaker CS stereotypes and increased positive attitudes toward teaching CS.

3. Purpose and Research Questions

The purpose of this study was to investigate if exposure to a stereotypical vs. counter-stereotypical role model affects teacher candidates' CS stereotypes and attitudes toward teaching CS. These questions guided the study:

RQ1: Does exposure to a counter-stereotypical role model affect teacher candidates' stereotypes about computer scientists?

RQ2: How does exposure to a counter-stereotypical role model affect teacher candidates' attitudes toward teaching computer science?

4. Methods

4.1 Research Design

This was a quasi-experimental study as it sought to determine the impact of an intervention on a target population that is not randomly assigned to experimental groups (Gopalan et al., 2020). Study participants were assigned to different groups based on course enrollment: participants in one group were exposed to a stereotypical role model and participants in another group were exposed to a counter-stereotypical role model. This study did not have a control group.

4.2 Setting and Participants

Participants were recruited from four sections of a teacher education course on early childhood mathematics teaching. The course was hybrid, i.e., it offered both online and face-to-face activities. Two sections of the course were offered in Fall 2020, and the same full-time female faculty taught them. The other two sections were offered in Fall 2021, and a male adjunct instructor taught both sections. Institutional Review Board approval was granted prior to the study. Informed consent was obtained. A total of 36 female senior teacher candidates agreed to join the study. Thirty-one were White, four were Latinx, and one was African American. Their average age was 21.81 years old (*SD* = 1.79). Participants were randomly assigned to either a counter-stereotypical (n = 15) or a stereotypical group (n = 21).

4.3 Instructional Material

Two versions of an online module were developed to correspond with the two role models, a counter-stereotypical and a stereotypical role model, in this study. The online module about STEM teaching and learning was designed and developed by the authors. This online module introduced participants to the idea of coding as a strategy to promote STEM learning, and they were prompted to reflect about integrating block-based code into their future STEM teaching. This module contained videos about STEM teaching and learning in early childhood, readings about the integration of coding into the classroom, and sample STEM activities (e.g., integrating coding into mathematics learning) for review. The role model was a computer scientist who guided teacher candidates through module activities and shared personal information throughout the module.

Group 1 was exposed to a counter-stereotypical role model, and group 2 to a stereotypical role model. The role models served as contextually-relevant pedagogical agents, which "are static or animated anthropomorphic interfaces employed in electronic learning environments to serve various instructional goals" (Veletsianos, 2010, p. 577). Role model avatars were designed with Bitmoji, a free avatar design tool. Bitmoji offers various scenarios in which the avatar displays emotions, preferences, and interactions.

To control stereotypicality, we designed role models based on five dimensions: race, gender, cognitive skills, social skills, and work preferences. These dimensions are often pointed out in the literature about CS stereotypes. First, we created the stereotypical role model: a White male who is highly intelligent, antisocial, and spends long hours working on the computer rather than around people. Other traits stereotypically associated with computer scientists were also featured such as glasses, preference for sci-fi movies (e.g., Star Wars) and video games. The counter-stereotypical role model was an African American female who did not mention having exceptional intelligence but was sociable and enjoyed spending time with friends. Traits that are not stereotypically associated with computer scientists were also featured such as a feminine outfit, and appreciation for TV shows (e.g., Friends) and movies. In addition to the avatar image, each CS role model was presented with written descriptions that reinforced stereotypical and counter-stereotypical features for the five dimensions mentioned above. We understand that, in reality, it is possible to hold different combinations of stereotypical vs. counter-stereotypical perceptions of a computer scientist. However, it was our goal for this study to assess the impact of these two role models in their extremeness. Figures 1 and 2 present a comparison of the two CS role models used in the experiment.

NERD ALERT! So now, a little bit about me. I'm 30 years old, and I was named after James Kirk, one of the Star Trek characters. My parents are huge fans! In my free time, I like playing online games, computer coding, playing chess, and watching fiction movies such as Star Wars and Lord of the Rings. I guess I can describe myself as an introvert who is a bit socially awkward. I usually spend a lot of the time on my computer. I'm excited about this module because it is about STEM and coding. I love coding, and I'm very good at it!

Figure 1. Stereotypical role model

So now, a little bit about me. I'm 30 years old, and I was named after Serena Williams, the famous tennis player. My parents are huge fans! In my free time, I spend a lot of time with friends. We go to concerts, the beach, and watch TV shows such as Friends and Grey's Anatomy. I guess I can describe myself as an extrovert because I love being social and surrounded by people. Besides being with friends, I usually spend some time on my computer lesson planning, but not a lot. I'm excited about this module because it is about STEM and coding. I love coding, and I'm very good at it! Here is a few tasks to start the module:

Figure 2. Counter-stereotypical role model

4.4 Data Sources and Analysis

To assess changes in teacher candidates' stereotypes, the CS Stereotypes Survey was administered before and after exposure to role models. This is an 18-item survey in which participants reported their perceptions of a computer scientist based on traits related to appearance, social skills, cognitive skills, and work preferences. This survey prompted participants to use a 5-point Likert-type scale to depict a computer scientist from (1) individualistic to (5) collaborative, or from (1) computer hacker to (5) amateur tech user. A rating value close to one indicates a stronger stereotypical belief about computer scientists, while a rating value close to five indicates a stronger counter-stereotypical belief. Survey data was analyzed with the nonparametric Mann-Whitney U test, which was suitable to examine if there were statistically significant differences between the two unrelated groups when the variable of interest is ordinal (Nolan & Heinzen, 2012; Siegel, 1956). This test is also appropriate for statistical analysis with relatively small samples, especially given the number of participants in group 1.

To assess changes in teacher candidates' attitudes toward teaching CS, an adapted version of Yadav et al.'s (2011) survey was implemented before and after exposure to role models. This Attitudes toward Teaching CS instrument was a 16-item survey that used a 4-point Likert type scale ranging from (1) strongly disagree to (4) strongly agree. Sample survey items include "I can do well in infusing coding and computing into teaching" and "Computer science and coding can be integrated into classroom education in other fields." Descriptive statistics were used to analyze this data set. A measure of central tendency (mean) and a measure of dispersion (standard deviation) provided trends and patterns

in collective data across participants (Field, 2017; Nolan & Heinzen, 2012) in their attitudes toward teaching CS.

Both surveys were piloted with teacher candidates prior to this study. Additionally, to assess the internal consistency of the CS Stereotypes Survey in this study, Cronbach's alpha coefficients were calculated for each subscale. The reliability scores were reported as .90 for social skills, .80 for appearance, .77 for cognitive skills, and .84 for work preferences, indicating good subscale reliability (Fraenkel & Wallen, 2009).

5. Findings

5.1 CS Stereotypes

Separate Mann-Whitney U tests were conducted to determine if exposure to a stereotypical vs. counter-stereotypical role model affects teacher candidates' stereotypical beliefs about computer scientists. In terms of computer scientists' social skills, teacher candidates in the counter-stereotypical role model group (*Mdn* = 0.60) reported a significantly higher decrease in their stereotypical beliefs compared to the teacher candidates in the stereotypical role model group (*Mdn* = 0), $U = 78.50$, $p < .01$, $r = 0.43$). Besides the social skills, teacher candidates in the counter-stereotypical role model group reported a decrease in their stereotypical beliefs about the appearance, cognitive skills, and work preferences of computer scientists. However, these changes were not statistically significantly different than the changes reported by participants from the stereotypical role model group in terms of appearance (U = 98.50, p = .05, r = 0.33), cognitive skills (U = 98.50, p = .22, r = 0.21), and work preferences (U = 116.00, p = .18, r = 0.23). Table 1 below presents the descriptive statistics for the change in teacher candidates' stereotypical beliefs about computer scientists in both groups.

Table 1. Descriptive statistics for the CS stereotypes (change score = post – pre)

	Counter-Stereotypical Role Model Group			Stereotypical Role Model Group		
	Mdn[a]	*M*	*SD*	*Mdn*[a]	*M*	*SD*
Social Skills	0.60	0.64	0.85	0	-0.08	0.57
Cognitive Skills	0.50	0.55	0.76	0.50	0.39	0.54
Appearance	0.50	0.50	0.97	0	0.06	0.36
Work Preferences	0.40	0.47	0.73	0	0.13	0.73

Note. [a]Higher values of *Mdn* indicate a decrease in stereotypical beliefs after completing the modules. Values close to zero indicate no change.

5.2 Attitudes toward Teaching CS

Participants in both groups showed increasingly positive attitudes toward CS and toward integrating CS into their teaching after the experiment. Regarding teacher candidates exposed to the stereotypical role model, notable increases were observed in their willingness to take CS courses (M = 2.87), the hope that their future career would require using coding and CS concepts (M = 2.67), the expectation to use coding in future education and professional career (M = 3.07), and their self-efficacy beliefs about infusing coding and computing into their teaching (M = 2.93). On the other hand, teacher candidates exposed to the counter-stereotypical role model experienced an increase in their expectation to use coding in their future education and professional career (M = 3.00), the perception that the challenge of teaching with coding is appealing (M = 2.81), their willingness to take CS courses (M = 2.81), and the perception that infusing coding and CS into teaching is interesting (M = 3.00). Table 2 below presents the descriptive statistics of teacher candidates' ratings for all items in the Attitudes toward CS Survey for both groups before and after the experiment.

Table 2. Descriptive statistics for attitudes toward teaching CS

Survey Item	Stereotypical Role Model Group		Counter-Stereotypical Role Model Group	
	Before M (SD)	After M (SD)	Before M (SD)	After M (SD)
1. Knowledge of coding will allow me to secure a better job as a teacher.	2.53 (0.52)	2.73 (0.88)	2.57 (0.68)	2.95 (0.74)
2. My teaching career goals do not require that I learn computing skills such as coding	2.73 (0.70)	2.40 (0.63)	2.67 (0.73)	2.29 (0.46)
3. I doubt that I can infuse coding or computing applications into my teaching.	2.13 (0.64)	1.93 (0.59)	2.10 (0.54)	1.81 (0.60)
4. I expect to use coding in my future educational and career work as a teacher.	2.33 (0.62)	3.07 (0.59)	2.33 (0.48)	3.00 (0.45)
5. I can do well in infusing coding and computing into teaching.	2.20 (0.68)	2.93 (0.80)	2.67 (0.66)	2.86 (0.57)
6. The challenge of teaching using computer science and coding appeals to me.	2.40 (0.63)	2.80 (0.86)	2.29 (0.64)	2.81 (0.60)
7. I expect to use coding and computer science for future teaching involving teamwork.	2.33 (0.62)	3.00 (0.65)	2.52 (0.60)	2.81 (0.51)
8. I can learn to teach coding and computing concepts.	2.80 (0.68)	3.13 (0.64)	3.00 (0.55)	3.19 (0.51)

9. I am not comfortable with teaching coding and computing concepts.	2.67 (0.82)	2.40 (0.74)	2.62 (1.02)	2.48 (0.68)
10. I expect to use coding and computing skills in my daily life as a teacher.	2.00 (0.66)	2.67 (0.90)	2.33 (0.66)	2.57 (0.60)
11. I hope that my future career as a teacher will require the use of coding and computing concepts.	1.80 (0.68)	2.67 (0.72)	2.43 (0.75)	2.76 (0.70)
12. I think that the idea of infusing coding and computer science into teaching is interesting.	2.47 (0.83)	3.07 (0.70)	2.57 (0.81)	3.00 (0.55)
13. I will voluntarily take computing courses if I were given the opportunity.	2.00 (0.76)	2.87 (0.64)	2.33 (0.86)	2.81 (0.51)
14. Computer science and coding can be integrated into classroom education in other fields.	3.00 (0.38)	3.20 (0.41)	2.86 (0.66)	3.19 (0.51)
15. Computer science and coding should be integrated into classroom education for other disciplines.	2.80 (0.56)	3.13 (0.52)	2.76 (0.54)	3.14 (0.48)
16. Having background knowledge and understanding of how to infuse computer science and coding into one's own teaching is valuable in and of itself.	3.13 (0.52)	3.33 (0.49)	3.00 (0.55)	3.19 (0.51)

6. Discussion and Future Research

The present study investigated if exposure to a stereotypical vs. counter-stereotypical role model influenced teacher candidates' stereotypical beliefs about CS and their attitudes toward teaching CS. Our hypothesis was that exposure to counter-stereotypical role models would result in weaker CS stereotypes and increased positive attitudes toward teaching CS. Study findings revealed that there were no statistically significant changes in CS stereotypes among teacher candidates exposed to the stereotypical role model. In fact, descriptive statistics showed that there were virtually no changes before and after the experiment. Participants in the counter-stereotypical role model group reported a statistically significant decrease in stereotypical beliefs about a computer scientist's social skills, and a slight increase in other dimensions. This partially aligns with previous research, which shows positive effects of counter-stereotypical role models (Cheryan, Siy, et al., 2011; Cheryan, Drury, et al., 2012; Cheryan et al., 2015; Stout et al., 2011) and environmental cues (Cheryan et al., 2009; Master et al., 2016) on females.

Participants in the counter-stereotypical group described computer scientists as more sociable and outgoing after the experiment. The counter-stereotypical role model was a female, and females are often attributed gender-role stereotypes based on societal expectations that they are sociable and

talkative (Block, 1973; Rosenkrantz et al., 1968). Using social role theory as the interpretive lens, it is possible that this gender-role stereotype overpowered the stereotype of a computer scientist being antisocial, resulting in teacher candidates' perception of a more sociable and extroverted female computer scientist. Studies about CS stereotypes among children have found that gender plays a role in CS steretoypes. Specifically, boys tend to show more interest in CS (Master et al., 2021), they are more commonly associated with the trait intelligence (Bian et al, 2017), they often display more positive attitudes toward CS (Vandenberg et al., 2021), and they are considered more capable in computer programming than girls (de Wit et al., 2022). This points to intersectionality in stereotypes as mental schemata that are influenced by various social constructs such as race, gender, sexuality, and more (Ireland et al., 2018; Rodriguez & Lehman, 2017; Trauth et al., 2016). While an intersectional analysis is beyond the scope of this study, we invite future research to adopt an intersectional theoretical framework to examine CS stereotypes.

Participants in the counter-stereotypical group externalized slightly weaker stereotypical beliefs about a computer scientist's appearance, cognitive skills, and work preferences though changes were not statistically significant. Participants in this study were overwhelmingly White, and the counter-stereotypical role model was African American. Perhaps participants did not "subjectively identify with" (Asgari et al., 2012, p. 371) the role model due to racial incongruence. This may partially explain the non-statistically significant difference about appearance in the counter-stereotypical group. Effective role models are most likely relatable (Asgari et al., 2012; Farrell et al., 2020; Shin et al., 2016) as they allow participants to build interpersonal connections and allow them to develop "a sense of perceived similarity to the role model" (Drury, Siy, & Cheryan, 2011, p. 267). Follow-up studies may include a number of counter-stereotypical role models that are contextually-relevant and demographically diverse in order to promote a sense of perceived similarity between teacher candidates and role models.

The experiment in this study was designed to represent the role model with images and text, which was expected to enhance content assimilation. According to the multimedia principle in Mayer's (2005) principles of multimedia learning, a combination of pictures and words leads to more effective learning rather than words alone. And yet, most changes were not statistically significant. We speculate that explicitly singling out and calling participants' attention to the five stereotype dimensions in the role models could have been more impactful. Specifically, we believe it would have been beneficial to combine the segmenting principle for multimedia learning (Mayer, 2005; Mayer & Pilegard, 2005) with cognitive scaffolding strategies (Belland et al., 2013) to explicitly challenge and offer alternatives to each one of the CS stereotype dimensions as well as offer opportunities for scaffolded reflection about each dimension. This recommendation can inform future studies.

From the perspective of social role theory (Wood & Eagly, 2012), stereotypes are formed through social interactions (Good et al., 2010), which are exchanges between individuals in a given social context. Research has found promising results from mediating brief social interactions, in person or virtually, with a human being who embodies a counter-stereotypical role model (Cheryan, Drury, et al., 2012; Riegle-Crumb et al., 2017). Further, an intervention that entailed reading stories about successful counter-stereotypical role models (e.g., successful females in STEM) to young girls (Gilbert, 2015) found positive results. Hindsight about the design of the present study shows that teacher candidates read information about the role model, but they did not get to exchange information with the role model. We conjecture that participants may need more substantial and extended interactions with a counter-stereotypical role model to experience change in their long-ingrained stereotypical beliefs. These serve as recommendations for follow-up research. Additionally, future research can examine if different types of interaction with a counter-stereotypical role model (e.g., reading about role models, watching role models, or engaging in group vs. one-on-one conversation) have different effects on participants' CS stereotypes.

Findings from this study also revealed that participants in both stereotypical and counter-stereotypical groups reported more positive attitudes toward teaching CS after the experiment. The mean scores that resulted from exposing teacher candidates to a stereotypical vs. counter-stereotypical role model were very similar. Descriptive statistics showed that increases occurred across most survey items, particularly those about willingness to take future CS courses and expectation to use coding in their future education and career. It is possible that exposure to a role model, regardless of stereotypicality, raised teacher candidates' awareness to the importance of infusing coding skills into their future students' learning experiences. Another plausible explanation to these findings is that the content of the online module affected both groups. The module in which the experiment was embedded presented teacher candidates with a video and a practitioner's article about integrating STEM into K-12 education. It is likely that the content of these artifacts had a cumulative effect in positively influencing teacher candidates' attitudes toward infusing coding and CS into their future teaching. We invite follow-up research that includes a control group that is not exposed to a role model and/or to content about coding and STEM education.

7. Implications for CS Teacher Education

Investigating teacher candidates' stereotypes and attitudes toward teaching CS is critical because these factors can thwart opportunities for CS education in K-12 learning environments. Study findings point to a few implications for future practice within teacher education programs. First, the study showed that teacher candidates hold stereotypical beliefs about computer scientists. It is critical to address and debunk these stereotypes in order to prepare educators who can offer inclusive CS educational opportunities. Second, interventions on stereotypes should do more than expose teacher candidates to

one counter-stereotypical role model. Promoting interactions with multiple and demographically diverse role models should enhance the effectiveness of future interventions. Third, sustained exposure to counter-stereotypical role models might prove to be more effective than one-shot encounters. While there is no consensus in the literature about a specific timeline, we hypothesize that extended interventions or interactions that span over multiple time points might yield statistically significant results. Fourth, interactions with role models should be followed by opportunities for scaffolded reflection so teacher candidates can have the time and space to externalize their perceptions and beliefs about each dimension that is relevant for CS stereotypes. And last but not least, future practice in teacher education programs should adopt an intersectional approach to illuminate the extent to which social expectations based on race and gender are reflected on CS stereotypes, and to show how teacher candidates can demystify these intersectional stereotypes in their future teaching.

8. Study Limitations

This study had four limitations. First, the number of participants in each group was unbalanced, but this was based on the number of participants who accepted to join the study and who completed both pre- and post-surveys. Second, one of the groups had a relatively small number of participants for statistical analysis, which informed our decision to use a nonparametric test. Third, it was not possible to identify if the difference in course instructors (full-time female professor vs. adjunct male professor) influenced the results. Fourth, the experiment was designed to be completed in one sitting, without interruptions. It was not possible to oversee participants' experiment completion. The experiment implementation had to occur online and asynchronously due to data collection restrictions imposed by the covid-19 pandemic.

Declaration of Interest Statement: The authors report there are no competing interests to declare.
Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

Ari, F., Arslan-Ari, I., & Vasconcelos, L. (2022). Early childhood preservice teachers' perceptions of computer science, gender stereotypes, and coding in early childhood education. *Tech Trends*. https://doi.org/10.1007/s11528-022-00725-w

Asgari, S., Dasgupta, N., & Stout, J. G. (2012). When do counterstereotypic ingroup members inspire versus deflate? The effect of successful professional women on young women's leadership self-concept. *Personality and Social Psychology Bulletin*, *38*(3), 370–383. https://doi.org/10.1177/0146167211431968

Belland, B. R., Kim, C., & Hannafin, M. J. (2013). A framework for designing scaffolds that improve motivation and cognition. *Educational Psychologist*, *48*(4), 243-270. https://doi.org/10.1080/00461520.2013.838920

Berg, T., Sharpe, A., & Aitkin, E. (2018). Females in computing: Understanding stereotypes through collaborative picturing. *Computers & Education*, *126*, 105–114. https://doi.org/10.1016/j.compedu.2018.07.007

Beyer, S. (2014). Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education*, *24*(2–3), 153–192. https://doi.org/10.1080/08993408.2014.963363

Bian, L., Leslie, S., & Cimpian, A. (2017). Gender stereotypes about intellectual ability emerge early and influence children's interests. *Science*, *355*(6323), 389–391. https://doi.org/10.1126/science.aah6524

Block, J. H. (1973). Conceptions of sex role: Some cross-cultural and longitudinal perspectives. *American Psychologist*, *28*(6), 512–526. https://doi.org/10.1037/h0035094

Buckley, C., Farrell, L., & Tyndall, I. (2021). Brief stories of successful female role models in science help counter gender stereotypes regarding intellectual ability among young girls: A pilot study. *Early Education and Development*. https://doi.org/10.1080/10409289.2021.1928444

Bureau of Labor Statistics, U.S. Department of Labor. (2021, April 15). *Occupational outlook handbook: Computer and information technology occupations*. U.S. Bureau of Labor Statistics. https://www.bls.gov/ooh/

Cheryan, S., Drury, B. J., & Vichayapai, M. (2012). Enduring influence of stereotypical computer science role models on women's academic aspirations. *Psychology of Women Quarterly*, *37*(1), 72–79. https://doi.org/10.1177/0361684312459328

Cheryan, S., Master, A., & Meltzoff, A. N. (2015). Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology*, *6*, 1–8. https://doi.org/10.3389/fpsyg.2015.00049

Cheryan, S., Meltzoff, A. N., & Kim, S. (2011). Classrooms matter: The design of virtual classrooms influences gender disparities in computer science classes. *Computers & Education*, *57*(2), 1825–1835. https://doi.org/10.1016/j.compedu.2011.02.004

Cheryan, S., Plaut, V. C., Davies, P. G., & Steele, C. M. (2009). Ambient belonging: How stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology*, *97*(6), 1045–1060. https://doi.org/10.1037/a0016239

Cheryan, S., Plaut, V. C., Handron, C., & Hudson, L. (2013). The stereotypical computer scientist: Gendered media representations as a barrier to inclusion for women. *Sex Roles*, *69*, 58–71. https://doi.org/10.1007/s11199-013-0296-x

Cheryan, S., Siy, J. O., Vichayapai, M., Drury, B. J., & Kim, S. (2011). Do female and male role models who embody STEM stereotypes hinder women's anticipated success in STEM? *Social Psychological and Personality Science*, *2*(6), 656–664. https://doi.org/10.1177/1948550611405218

de Wit, S., Hermans, F., & Aivaloglou, E. (2021). Children's implicit and explicit stereotypes on the gender, social skills, and interests of a computer scientist. *Proceedings of the 17th ACM Conference on International Computing Education Research*, 239–251. https://doi.org/10.1145/3446871.3469753

Drury, B. J., Siy, J. O., & Cheryan, S. (2011). When do female role models benefit women? The importance of differentiating recruitment from retention in STEM. *Psychological Inquiry*, *22*(4) 265–269. https://doi.org/10.1080/1047840X.2011.620935

Eagly, A. H., & Karau, S. J. (2002). Role congruity theory of prejudice toward female leaders. *Psychological Review*, *109*(3), 573–598. https://doi.org/10.1037//0033-295X.109.3.573

Eagly, A. H., Wood, W., & Diekman, A. B. (2000). Social role thoery of sex differences and similarities: A current appraisal. In T. Eckes & H. M. Trautner (Eds.), *The developmental social psychology of gender* (pp. 123–174). Erlbaum.

Farrell, L., Nearchou, F., & McHugh, L. (2020). Examining the effectiveness of brief interventions to strengthen a positive implicit relation between women and STEM across two timepoints. *Social Psychology of Education*, *23*, 1203-1231. https://doi.org/10.1007/s11218-020-09576-w

Field, A. (2017). *Discovering statistics using IBM SPSS statistics (5th ed.)*. SAGE Publications.

Fraenkel, J., & Wallen, N. (2009). *How to design and evaluate research in education* (7th ed.). Boston: McGraw-Hill.

Gilbert, P. (2015). *The role of role models: How does identification with STEM role models impact women's implicit STEM stereotypes and STEM outcomes? [Unpublished doctoral dissertation/master's thesis]*. Tulate University.

Good, J. J., Woodzicka, J. A., & Wingfield, L. C. (2010). The effects of gender stereotypic and counter-stereotypic textbook images on science performance. *The Journal of Social Psychology*, *150*(2), 132–147.

Gopalan, M., Rosinger, K., & Ahn, J. B. (2020). Use of quasi-experimental research designs in education research: Growth, promise, and challenges. *Review of Research in Education*, *44*(1), 218–243. https://doi.org/10.3102/0091732X20903302

Graham, S., & Latulipe, C. (2003). *CS girls rock: Sparking interest in computer science and debunking the stereotypes*. 322–326.

Ireland, D. T., Freeman, K. E., Winston-Proctor, C. E., DeLaine, K. D., Lowe, S. M., & Woodson, K. M. (2018). (Un)hidden figures: A synthesis of research examining the intersectional

experiences of black women and girls in STEM education. *Review of Research in Education*, *42*(1), 226-254. https://doi.org/10.3102/0091732X18759072

Kanahara, S. (2006). A review of the definitions of stereotype and a proposal for a progressional model. *Individual Differences Research*, *4*(5), 306–321.

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology*, *108*(3), 424–437. http://dx.doi.org/10.1037/edu0000061

Master, A., Meltzoff, A. N., & Cheryan, S. (2021). Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *Proceedings of the National Academy of Sciences*, *118*(48), 1–7. https://doi.org/10.1073/pnas.2100030118

Mayer, R. E. (2005). Cognitive theory of multimedia learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 31–48). Cambridge University Press. https://doi.org/10.1017/CBO9780511816819.004

Mayer, R. E., & Pilegard, C. (2014). Principles for managing essential processing in multimedia learning: Segmenting, pre-training, and modality principles. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 316–344). Cambridge University Press. https://doi.org/10.1017/CBO9781139547369.016

National Science Foundation. (2019). *Women, minorities, and persons with disabilities in science and engineering: 2019*(Special Report NSF 19-304). https://www.nsf.gov/statistics/wmpd

Nolan, S. A., & Heinzen, T. E. (2012). *Statistics for the behavioral sciences (2nd ed.)*. Worth Publishers.

Olsson, M., & Martiny, S. E. (2018). Does exposure to counterstereotypical role models influence girls' and women's gender stereotypes and career choices? A review of social psychological research. *Frontiers in Psychology*, *9*, 1–15. https://doi.org/10.3389/fpsyg.2018.02264

Pantic, K., Clarke-Midura, J., Poole, F., Roller, J., & Allan, V. (2018). Drawing a computer scientist: Stereotypical representations or lack of awareness? *Computer Science Education*, *28*(3), 232–254. https://doi.org/10.1080/08993408.2018.1533780

Papadakis, S. (2018). Gender stereotypes in Greek computer science school textbooks. *International Journal of Teaching and Case Studies*, *9*(1), 48–71. https://doi.org/10.1504/IJTCS.2018.10011123

Riegle-Crumb, C., Moore, C., & Buontempo, J. (2017). Shifting STEM stereotypes? Considering the role of peer and teacher gender. *Journal of Research on Adolescence*, *27*(3), 492–505. https://doi.org/10.1111/jora.12289

Rodriguez, S. L., & Lehman, K. (2017). Developing the next generation of diverse computer scientists: the need for enhanced, intersectional computing identity theory. *Computer Science Education*, *27*(3-4), 229-247. https://doi.org/10.1080/08993408.2018.1457899

Rosenkrantz, P., Vogel, S., Bee, H., Broverman, I., & Broverman, D. M. (1968). Sex-role stereotypes and self-concepts in college students. *Journal of Consulting and Clinical Psychology*, *32*(3), 287–295. https://doi.org/10.1037/h0025909

Shapiro, J. R., & Williams, A. M. (2012). The role of stereotype threats in undermining girls' and women's performance and interest in STEM fields. *Sex Roles*, *66*, 175–183. https://doi.org/10.1007/s11199-011-0051-0

Shin, J. E. L., Levy, S. R., & London, B. (2016). Effects of role model exposure on STEM and non-STEM student engagement. *Journal of Applied Social Psychology*, *46*(7), 410–427. https://doi.org/10.1111/jasp.12371

Siegel, S. (1956). *Non-parametric statistics for the behavioral sciences*. McGraw-Hill.

Sills, D. L. (1968). *International encyclopedia of the social sciences*. The Macmillan Company & The Free Press.

Stout, J. G., Dasgupta, N., Hunsinger, M., & McManus, M. A. (2011). STEMing the tide: Using ingroup experts to inoculate women's self-concept in science, technology, engineering, and mathematics (STEM). *Journal of Personality and Social Psychology*, *100*(2), 255–270. https://doi.org/10.1037/a0021385

Taylor, S. E., Peplau, L. A., & Sears, D. O. (1994). *Social psychology*. Prentice Hall.

Trauth, E. M., Cain, C. C., Joshi, K. D., Kvasny, L., & Booth, K. M. (2016). The influence of gender-ethnic intersectionality on gender stereotypes about IT skills and knowledge. *The Data Base for Advances in Information Systems*, *47*(3), 9-39. https://doi.org/10.1145/2980783.2980785

Vandenberg, J., Rachmatullah, A., Lynch, C., Boyer, K. E., & Wiebe, E. (2021). Interaction effects of race and gender in elementary CS attitudes: A validation and cross-sectional study. *International Journal of Child-Computer Interaction*, *29*, 1-11. https://doi.org/10.1016/j.ijcci.2021.100293

Varma, R. (2020). Women in computing education: A Western or a global problem? Lessons from India. In C. Frieze & J. L. Quesenberry (Eds.), *Cracking the digital ceiling: Women in computing around the world* (pp. 299–310). Cambridge University Press. https://doi.org/10.1017/9781108609081.018

Vasconcelos, L., Ari, F., Arslan-Ari, I., & Lamb, L. (2022). Female preservice teachers stereotype computer scientists as intelligent and overworked White individuals wearing glasses. *Computers & Education*, 187. https://doi.org/10.1016/j.compedu.2022.104563.

Veletsianos, G. (2010). Contextually relevant pedagogical agents: Visual appearance, stereotypes, and first impressions and their impact on learning. *Computers & Education*, *55*(2), 576-585. https://doi.org/10.1016/j.compedu.2010.02.019

Wood, W., & Eagly, A. H. (2012). Biosocial construction of sex differences and similarities in behavior. In J. M. Olson & M. P. Zanna (Eds.), *Advances in Experimental Social Psychology* (Vol. 46, pp. 55–123). Academic Press. https://doi.org/10.1016/B978-0-12-394281-4.00002-7

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 465–470. http://dl.acm.org/citation.cfm?id=1953297

Core Competencies of K-12 Computer Science Education from The Perspectives of College Faculties and K-12 Teachers

Meina Zhu[1]

Cheng Wang[1]

[1]Wayne State University, USA

DOI: https://doi.org/10.21585/ijcses.v6i2.161

Abstract

Given the increasing need for employees with computational skills, understanding the core competencies of K-12 computer science (CS) education is vital. This phenomenological research aims to identify critical factors of CS education in K-12 schools from the perspectives and visions of CS faculties in higher education and teachers in K-12 schools. This study adopted a phenomenological research design. The researchers conducted a semi-structured interview with 13 CS faculties and K-12 CS teachers in Michigan and analyzed the data using thematic analysis. The findings indicated that: (1) the core competencies for K-12 CS education include problem-solving through computational thinking, math background, and foundational programming skills, and (2) what is essential is not the programming languages taught in K-12 schools but computational thinking, which enables the learners to easily transfer from one language environment to another. The findings provide important implications for K-12 CS education regarding the core competencies and programming languages to be taught.

Keywords: K-12 computer science education, core competencies, computational thinking, problem-solving, math

1. Introduction

As computers become one of the essential social fabrics that construct the infrastructure of our world, the need for K-12 computer science (CS) education is increasing. The CS education community made K-12 CS education standards in 2017 which “delineate a core set of learning objectives designed to provide the foundation for a complete computer science curriculum and its implementation at the K-12 level” (CSTA, n.d.). For each state, defining CS and establishing rigorous K-12 CS standards is one of the nine policies to be developed according to the Code.org advocacy coalition. Michigan adopted the Computer Science Teachers Association (CSTA) K-12 CS standards in 2019 (Code.org, CSTA, & ECEP Alliance., 2020). However, only 37% of Michigan high schools offered CS courses during the

2019-2020 academic year (Michigan Department of Education, 2020). A majority of schools do not have a clear understanding of CS education and its needs, which may hinder their adoption and implementation of CS education. Given that CS faculty in higher education usually hold a doctoral degree in the field and have in-depth knowledge about CS education, their perceptions of core CS competencies and expectations from high school graduates can provide insights into K-12 CS education. At the same time, K-12 CS teachers are the practitioners in the field, and thus their experiences and feedbacks are as important as that of CS faculties in higher education. Therefore, this study aims to identify key factors in pre-college CS education from the perspectives and visions of CS college faculties and K-12 CS teachers so that CS researchers, educators, experts, policymakers, and other stakeholders in the field can provide better K-12 CS education to students.

2. Literature Review

2.1 K-12 CS Education

Given the importance of computing technology in modern society, the needs of employees with CS skills were increasing (Barr & Stephenson, 2011). CS has been widely adopted in diverse scientific and humanity areas. Nowadays, scientific and research innovations in social and humanity areas could not be accomplished without computers or computing skills (Gal-Ezer & Stephnson, 2014). Thus, CS knowledge and skills become essential in the 21st century.

CS was defined as the area that studies computers and algorithms, such as principles, hardware, and software design, applications, and evaluation by the Association for Computing Machinery (ACM) and the Computer Science Teachers Association (CSTA) K-12 standards task force (Seehorn et al., 2016). CS education in K-12 settings can develop students' higher-order thinking skills, reflective thinking skills, and critical thinking skills (Tran, 2019) for problem-solving (Ministry of Education, 2014).

K-12 CS education has been implemented in several countries. For example, Webb et al. (2017) investigated K-12 CS education curricula in five counties and found that these countries have agreed on the importance of CS and the advantages of having CS education as early as possible in K-12. However, there are still multiple concerns regarding K-12 CS education. The very first one is whether it is necessary to teach K-12 students CS since not all students will pursue CS majors or careers in the future (Grover & Pea, 2013). Next, if K-12 CS education is necessary, what are the core competencies to be developed among students? Lastly, given that curricula in K-12 is already packed and the time and space for CS education is limited, which kinds of programming languages and environments will be more appropriate and effective in implementation?

2.2 Problem Solving and Computational Thinking in CS Education

One of the primary purposes of CS is to solve computational problems. The problem-solving approach is often related to computational thinking (CT) (Grover & Pea, 2013; Israel et al., 2015), which has long been considered as one of the key factors in CS education. CT refers to using an algorithmic approach to solve real-world problems, which is a necessary skill in different contexts and situations (Shute et al., 2017). The term, CT, was introduced by Seymour Papert's book (1980) regarding the programming language LOGO. Later, Wing (2006) defines CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). Wing (2006) considers CT just as one of the analytical abilities like reading, writing, and arithmetic. Since 2006, CT has become a popular term in the CS education field. Regarding the components of CT skills, Selby and Woollard (2013) define CT as five subcomponents models: abstraction, decomposition, algorithm, generalization, and evaluation. While the definitions of CT were inconsistent and vague (Korkmaz et al., 2017), there is a common understanding of CT education: with CT skills, students can think like CS professionals to solve problems through steps such as decomposition, pattern recognition, and algorithm (Barr & Stephenson, 2011).

Give its values in modern society, CT is considered not only as one of the skills that could change students' thinking in different fields (Papert, 1980) but a universal skill for every student to obtain (Barr & Stephenson, 2011; Voogt et al., 2015). The OECD and UNESCO state that CT is a necessary skill for digital citizens (Organisation for Economic Cooperation and Development, 2018; World Economic Forum, 2015). International Society for Technology in Education (ISTE) (2018) has included CT as one of the learning standards so that students can use computational methods to solve problems in the digital era. Moreover, CT is proposed to be included in compulsory education in the report from European Commission (Bocconi et al., 2016). Thus, some countries have included CT in their curricula, such as the U.K. and Australia (Bower et al., 2017).

2.3 Programming Languages in CS Education

Programming is one of the fundamental skills in CS and a vital tool to develop CT skills (Grover & Pea, 2013; Lye & Koh, 2014). Research indicated that introducing CT to students in their earlier years is important as it could equip students with critical thinking skills (Tran, 2019). The programming approach has been implemented for CT education in pre-school (e.g., Çiftci & Bildiren, 2020) and K-12 education (e.g., Schmidt, 2016). For example, Çiftci and Bildiren (2020) found that programming can help develop 4-5-year-old preschool students' problem solving and cognitive skills. Irish and Kang (2018) found that integrating programming into other learning activities can engage students in both programming and general subjects learning.

Consequently, programming languages and environments play an important role in K-12 CS education. The question of which language should be taught in K-12 has been a controversial topic. Currently, popular programming languages such as Python, Java, C, and C++ are widely used in industry and academia (TIOBE, 2021). These languages are also called textual programming languages as they are primarily written in text editors. Therefore, programmers should learn not only logical thinking but also the syntax of the language. Although textual programming languages may be difficult to approach for novice learners, research has indicated that students who learn textual programming language as the introductory programming language can transit to other textual languages easier as they move forward (Enbody & Punch, 2010). Thus, they recommend that it is preferable to have textual programming language for novice learners, given that the textual programming languages are universally used in real life.

On the contrary, the non-textual programming languages and environments, which comprises diverse visual formats such as diagrams, flowcharts, and coding blocks (Dehouck, 2016), are expected to be easy enough for beginners to get started and extensive enough to meet the needs for advanced programming (Grover & Pea, 2013). Visual programming environments that are widely used include Scratch, Game Maker, Code.org, Alice, Kodu, etc. Some of the visual programming languages, such as Scratch and Alice, are block-based languages in the programming environments, of which students can drag and drop coding blocks to the workspace. Thus, novice CS learners can focus on the computational concepts and logic without being bothered by the syntax (Bau et al., 2017; Kelleher & Pausch, 2005). Some research argues that visual programming languages might be more appropriate for novice learners as they are easier to learn (Bau et al., 2017; Chen et al., 2020; Malan & Leitner, 2007). For example, Chen et al. (2020) analyzed data from 10,000 undergraduate students who enrolled in CS courses and found that students whose first programming language was visual performed better than did students whose first programming language was textual when the programming languages were first introduced in K-12 stages. Moreover, the visual languages and environments provide scaffolds and enable knowledge transfer. Research indicates that visual programming languages are used in K-12 CT education. For example, Hsu et al. (2018) and Lockwood and Mooney (2017) find that many schools have utilized visual programming languages to teach CT skills. Other studies indicate that using visual programming languages to teach students CT skills is effective in elementary education (The Horizon Report, 2017). Application of visual program languages in K-12 CS education is found to significantly improve students' understanding of computational concepts and computation practices (Saez-Lopez et al., 2016), logical thinking skills (Lindh and Holgersson 2007), and problem-solving skills in general (Chou 2018).

Given the mixed opinions of introducing visual or textual programming languages in K-12 CS education, Xu et al. (2019) conduct a meta-analysis on the block-based versus textual programming on student learning outcomes by reviewing 13 publications. They find a small effect size in favor of block-based programming languages on cognitive learning outcomes and suggest more research on the effectiveness of using block-based programming languages for novice learners in the future.

To sum up, the epistemology of K-12 CS education, including its necessity, its core competencies as well as programming languages that should be taught in its implementation, still need to be clarified. Thus, this study aims to explore the K-12 CS students' core competencies and programming languages that should be learned in K-12 from CS professors' and K-12 teachers' perspectives.

The following research questions guide this study:

(1) What are the CS competencies expected from K-12 students from the perspectives of CS faculties in higher education and teachers in K-12 schools?

(2) What are the programming languages to be introduced to K-12 students from the perspectives of CS faculties in higher education and teachers in K-12 schools?

3. Method

To answer the two research questions, we use the qualitative interview data coming from the Computer Science Teachers in Michigan (CSTIM) project that led by the two authors of the present study. The CSTIM project adopts a mixed-method design (Creswell & Plano-Clark, 2017) to investigate the necessity of K-12 CS education, core competencies of CS learners, current trends and issues related to K-12 CS education, and teaching strategies as well as teachers' competencies to teach CS in K-12 schools. The project is comprised of three components. First, from the ideological perspective, the researchers aim to capture the fundamental values in CS education and the core competencies for K-12 CS students through semi-structured interviews of CS college faculties and K-12 CS teachers. Second, from the practical perspective, the researchers investigate the CS teaching strategies, K-12 teacher competencies, and professional development approaches through semi-structured interviews. Third, based on the analysis results of the first two phases, the researchers extract the keywords for mining data from Twitter to examine the current trends and issues related to K-12 CS education. This current study focuses on the first component of our entire CSTIM project.

In its qualitative part, the CSTIM project applies a phenomenological research design (Giorgi & Giorgi 2003). In particular, we conduct semi-structured interviews with eight CS faculties in higher education and five CS teachers in K-12 schools to understand CS education phenomenon. We choose the qualitative approach was because it can benefit the discovery and interpretation of the investigated

phenomena (Yu & Hai, 2005). Moreover, the semi-structured interviews provide rich information about respondents' experiences and perceptions of CS education.

3.1 Instruments

An interview protocol can provide a framework to guide the semi-structured interviews (Patton, 2015). The interview protocol is developed from the literature review regarding K-12 CS education (i.e., CSTA n.d.; K-12 Computer Science Framework Steering Committee, 2016; Wing, 2006). The interview protocol for higher education CS faculties includes 13 questions related to CS learners' competencies, challenges, strategies, and expectations in K-12 CS education (Zhu & Wang, 2023). Please see the detailed interview protocol in Appendix. The first question is about the interviewee's background information. Questions two to seven are related to interviewees' perceptions of CS learners' competencies, programming languages, effective strategies, and challenges while teaching CS students and typical successful CS learners. Questions eight to ten are related to interviewees' opinions of the necessity of K-12 CS education, curricular, and programming languages. Question 11 to 13 are about K-12 CS teachers' competencies to teach K-12 CS courses.

The interview protocol for K-12 teachers includes 11 questions regarding their understanding of K-12 CS standards and competencies, experiences, and feedbacks in K-12 CS education, contents, and programming languages they used in classrooms (Zhu & Wang, 2023). The first question is about the interviewee's background information. Questions two to three are about their understanding of CS standards and CS education. Questions four to seven are related to K-12 CS learners' competencies, curricular, programming language, assessment approach. Questions eight to question 11 are related to K-12 CS teachers' teaching challenges, resources and support, and professional development. Given that this study adopts a semi-structured interview method, follow-up questions are asked based on each individual interviewee's response.

3.2 Participants

The participants of the CSTIM project include both faculties in higher education and K-12 teachers. The criteria for selecting the faculties in higher education include: (1) having at least three years' CS teaching experience, (2) have taught undergraduate freshman or sophomore courses, and (3) their universities are located in Michigan state. The criteria for choosing K-12 teachers are: (1) having experience of teaching CS courses in the past three years and (2) their schools are located in Michigan state. The researchers gather CS college faculties' emails from their university websites and send an email invitation to participate in our study. Eight CS college faculties accept the invitation and participate in the study. They come from six universities in Michigan, including the University of

Michigan, Wayne State University, Oakland University, Central Michigan University, Western Michigan University, and Eastern Michigan University. Seven out of eight CS instructors held a Ph.D. degree in CS, and one was working on his Ph.D. degree. To recruit K-12 CS teachers, the researchers use a snowball sampling method, and five K-12 CS teachers accept the invitation and participate in our study. The five interviewees include three high school teachers and two middle school teachers. Among the five teachers, only one had a bachelor's degree in CS. The rest of them did not have CS related degrees. Detailed information about the interviewees is shown in Table 1.

Table 1. Participant information

Pseudonyms	Occupations	Institutions	Educational background	Gender
Arthur	Teacher	High school	Ph.D. in physics	Male
Diego	Teacher	High school	Bachelor in CS & Master's degree in arts and teaching	Male
Eli	Teacher	Middle school	N/A	Male
Kate	Teacher	High school	CS workshops	Female
Lucy	Teacher	Middle and high school	Bachelor with a math major and CS minor; master's degree in teaching	Female
Aiden	Associate Professor	Higher education	Ph.D. in CS	Male
Daxton	Instructor	Higher education	Working on a Ph.D. degree in CS	Male
David	Associate Professor	Higher education	Ph.D. in CS	Male
James	Professor	Higher education	Ph.D. in CS	Male
Kash	Associate Professor	Higher education	Ph.D. in CS	Male
Lawrence	Associate Professor	Higher education	Ph.D. in CS	Male
Luke	Assistant Professor	Higher education	Ph.D. in CS	Male
Tong	Assistant Professor	Higher education	Ph.D. in CS	Male

3.3 Data Collection Procedures

The interview protocol is shared with the interviewees at least one day before the interview for them to prepare for the answers. Each interview lasts approximately 30 minutes. Since the CSTIM project is conducted during an ongoing pandemic of COVID-19, the face-to-face interview is infeasible. The interviews are primarily audio-recorded via Zoom, an online conference tool, along with the Smart Recorder app installed on the researchers' smartphone as a secondary means to secure the data

collection. The recordings are transcribed verbatim. To appreciate their participation, the researchers provide a $25 Amazon gift card after each participant validates his or her interview data.

3.4 Data Analysis

The researchers use thematic analysis (Braun & Clarke, 2006) to analyze the interview data. The thematic analysis enables researchers to identify patterns across datasets in order to describe the invested phenomenon (Guest, 2012). It includes six phases for researchers to form themes from the qualitative data (Bernard & Ryan, 2009). The first phase includes familiarizing with the data. Researchers read the data repeatedly to identify the patterns in the data. In the second phase, codes are generated by labeling words, phrases, sentences, and paragraphs. In the third phase, closely related codes are combined into themes. Fourth, the themes are reviewed and revised. Some themes might be grouped together, while others might be split. In the fifth phase, themes are defined and named. Finally, the results are reported.

In the present study, two researchers independently conduct the first five phases of the thematic analysis. Then we meet to discuss the individual analysis results. The discrepancies are discussed until we reach a consensus. The final coding scheme on K-12 CS educational ideology includes two concepts, i.e., K-12 CS competencies and K-12 programming languages (see Table 2).

Table 2. Coding themes

Theme	Concept	Code
K-12 CS educational ideology	K-12 CS competencies	Problem-solving with computational thinking
		Math background
		Foundational programming skills
	K-12 recommended programming languages	From block-based visual programs to syntax-based language
		Python, Java, C++
		Specific language does not matter

3.5 Trustworthiness

Several strategies are used to ensure the trustworthiness of the study, such as credibility, dependability, transferability, and confirmability (Lincoln & Guba, 1985). First, credibility refers to what extent the data reflect the 'truth' of the phenomenon (Erlandson et al., 1993). In the present study, first-level member validation is conducted with all the interviewees to verify the accuracy of the transcripts. Among the 13 interviewees, 12 participants confirm the transcripts or make minor

revisions. One participant does not respond to our request. Second, dependability refers to the replicability of the research in the same or similar contexts (Erlandson et al., 1993). This study ensures dependability by recording the procedures and problems of the project in documents. Third, transferability represents to what extent the study findings can be applied in other different contexts (Erlandson et al., 1993). In this study, a thick description of the research context, participants, and results is provided. Fourth, confirmability refers to the extent of avoiding biases (Erlandson et al., 1993). The present study documents all the research processes to make sure the original data sources can be traced back.

4. Findings

Regarding the context of this study, 12 out of 13 interviews believe that CS education is necessary for K-12 schools. The only exception is James, a professor in higher education, who thinks that math is better than CS to cultivate problem-solving skills and CT (at least for kindergarteners through to the eighth graders), and it is not the best way to force the students to learn CS which will bring burden to them.

Turning to the first research questions, thematic analysis results of the interview data related to K-12 CS ideology include two primary concepts: K-12 CS competencies and K-12 recommended programming languages. The following section will present each concept and code in detail.

4.1 Concept I: K-12 CS Competencies

The data analysis results in three primary codes – problem-solving with CT, math background, and foundational programming skills – that help construct the concept of K-12 CS competencies.

4.1.1 Code I: Problem-solving with Computational Thinking

11 out of the 13 interviewees emphasize that the core CS competency of K-12 students is problem-solving with CT. The data analysis results indicate that the skills of solving real-world problems are expected from CS students at all levels. For example, Aiden shares his opinions regarding the importance of problem-solving skills for CS students in general:

> "These things [hot fields in CS] go through cycles. Things that are hot today will not be hot tomorrow. So, a good way to prepare students is to give them this core competency so that they have really competent, independent, fundamental ideas of computer science, which is how the problem can be solved using our computing systems." (Aiden, a CS associate professor)

In particular, for K-12 students, problem-solving is considered as one of the core competencies in CS education as well. K-12 students are expected to master the core knowledge and skills in CS subjects. In addition, problem-solving skill is not only important for CS learning but also critical for learning in other subjects. For example, Eli, a K-12 CS teacher, expresses his opinion on CS education and highlights the importance of "solving problems and come up with solutions." Similarly, Kash emphasizes the importance of problem-solving skills in K-12:

> "I think at high school, instead of teaching them programming, it's better to teach them problem-solving because learning syntax is not a big deal. Whoever has dwelled more problem-solving skills are more successful because the fundamental concept of programming languages is the same. So, if we are building a problem-solving skill at high school, just teach them to have one simplest language, Python, that is more than enough rather than introducing too many programming languages." (Kash, a CS associate professor)

The approaches to solving problems vary. Among different approaches for problem-solving, in CS education, CT is one of the important methods. Five interviewees explicitly state that CT is an essential approach for problem-solving. Other interviewees implicitly explain the importance of CT without using the specific term CT. For example, Lawrence shares his opinions of CT and problem solving:

> "I feel like there's an advantage in students being exposed to computational thinking of solving a problem. When I say computational thinking, I mean solving a problem. The way that you do it computationally is to break it down into steps and solve it step by step. I think that's a little different from the kind of problem-solving techniques you learned in the other fields." (Lawrence, a CS associate professor)

Despite that the CT concept is used in CS education, as mentioned earlier, the definition and meaning of CT have not reached a consensus. CS educators have some fundamental understanding of CT. David, a CS associate professor, explained his understanding about CT "it's more like how to know, solve the problem using a computer, basically." And Aiden, explains his understanding of CT:

> "For this computational thinking, first of all, they need to develop some awareness whenever they encounter a problem. Once they have an awareness, the next step is to develop a mindset that problems can be solved using a computer so that it becomes second nature. When they encounter a new problem, they think I can do this, and then try to formulate some real solutions and maybe even develop basic programming skills for high school students." (Aiden, a CS associate professor)

In addition, CT is considered an important approach for problem-solving no matter whether the students will pursue a CS major in higher education or not after high school. Interviewees think that in

real life, CT is helpful for people who work in different fields. For instance, Lucy and Lawrence express their thoughts on CT:

> "I think it's incredibly useful. These are skills that go beyond just the computer science field, but in everyday life in any field. They're going to understand how to break down a problem, how to work on the solution, and how to design something to be a solution for some tasks. This is incredibly important." (Lucky, a K-12 CS teacher)

> "I think it'd been exposed to computational thinking is valuable in the same way that students take chemistry in high school… I still like every basic knowledge about the world and how it works, and the scientific method is valuable. I think having some idea about how computational things work and how to do computational problem solving is useful. I think a lot of students are going to have to use computation later in life. So, these are useful skills for them." (Lawrence, a CS associate professor)

4.1.2 Code II: Math Background

Seven out of 13 interviewees highlight the importance of math background and consider math as the key cornerstone of CS education in both K-12 and higher education settings. One of the interviewees, Eli, states "Computer Sciences is another language, but it's inherently about. I mean, it's mathematical, it's algorithmic it's breaking things apart in baby steps. And then figuring out the variety of options." (Eli, a K-12 Middle school CS teacher). Similarly, James says, "but of course, learning, you know, studying math, learning math is key. Critical to good computational thinking." (James, a CS professor). Kash further emphasizes the importance of math:

> "From here, we have, you know, a clue that this guy is more fit for IT, but a person who has done some programming and has solved problems is really good at mathematics, so did this [being good at mathematics] is at least a clue for parents as well as, you know, the candidate themselves that they are maybe a better fit for, you know, computer science. So, I think teachers first need to focus on this thing." (Kash, a CS associate professor)

Despite that math is considered one of the foundational subjects in CS education, not all CS students have sufficient knowledge for CS learning. Six out of the 13 interviewees mention that a common challenge for some CS students is that they lack a math background. Per David, "as I said that they have to learn how to think computationally and solve problems. And that's the difficult part, and that requires a lot of math background." (David, a CS associate professor). He further explains:

> "I think the main issue is that the students that select especially at our university, that choose to go in computer science, they select the major but lack the appropriate background. So, they have, you know, are having a hard time, you know, with their first classes like the data structures,

> especially those that are used a lot [in other CS courses]. So, their math background is very poor, and they struggle with that. So that's one big challenge…misconception is very, you know, damaging in a way because they are disappointed because they think that they just have to learn the language, but that's just a tool, as I said to them, they have to learn how to think computationally and solve problems. And that's the difficult part, and that requires a lot of math background and upgrades, and they said [those are] the classes they avoid anyway so [in the past]." (David, a CS associate professor)

4.1.3 Code III: Foundational Programming Skills

Besides math, a few interviewees think another important component of K-12 CS education is programming skills. For example, Aiden states, "it's like building a foundation, a strong foundation of CS core competency comprising things like programming." (Aiden, a CS associate professor) In addition, Lucy says, "I think the goal that we're hitting on for middle to upper school has been programming and building algorithms, debugging, breaking down code." (Lucy, a K-12 Middle school CS teacher) Students without foundational programming skills usually encounter setbacks when they enter college, as elaborated by Lawrence:

> "So, about half of our students coming to our program are coming from community colleges, are transferring from some other colleges. And about half of this. I mean, it's every year. It's almost exactly 50%. It's been that way for several years, um, and about half of them are first-time [CS] students…I tell students that, you know, if this is your first time, you know, taking a programming course, you know, other people maybe have more experience than you. That doesn't mean that they're better at doing this, and you are right. This means that, you know, they've been doing it longer. So, I think sometimes students get discouraged if, either this is my hypothesis, they get discouraged if they see that it's easy for some students and it's hard for them, but it might be easy for the other students because they've already, like you said, taken it in high school. I don't know the exact numbers, but we definitely have a reasonable number of students who do not have any real exposure to programming before they join our program. But we also have students who have taken programming before in high school." (Lawrence, a CS associate professor)

4.2 Concept II: K-12 Recommended Programming Languages

To cultivate computational thinking for problem-solving, our interviewees also express their epistemology about the programming languages that might be used in K-12 education to serve this specific purpose. The section below demonstrates three code categories regarding programming teaching programming languages in K-12 CS education: (1) from block-based programming to syntax-

based languages; (2) syntax-based languages: Python, Java, C++; and (3) specific programming language does not matter.

4.2.1 Code I: From Block-Based Programming to Syntax-Based Languages

Regarding specific programming languages that should be taught in K-12, eight out of 13 interviewees suggest starting from block-based visual programming tools, such as Scratch and code.org. For example, Aiden says, “so something like scratch will be very effective for young children. As for these young children, say grade six or below this kind of range. The priority should be about engagement, making it fun for them so they can see the problems can be solved for older children like high school children, then yes, absolutely.” (Aiden, a CS associate professor) Eli, a K-12 teacher, says, “I used code.org or scratch. That's all block-based programming. I want something to be manageable or something to be user-friendly, and I want whenever they come up with a solution.” Similarly, Kate, Lucy, Arthur, and Diego echo the idea of using Block-based programming tools to teach K-12 students CS subjects.

In addition, four interviewees also mention that it might be better to start with block-based visual programming tools, such as Scratch, then transit to syntax-based programming languages, such as Python and other languages. For example, David says,

> “I would say that if you start with a simple [programming language]. For elementary school, you have to choose something graphical. There are a lot of environments out there, like maybe Scratch and Alice, and there are a lot of others. And as you go up, let's say, middle school, you can start introducing nonvisual programming environments. And you can go, you know, it doesn't really matter, if Java or Python or C++ will be more difficult to learn, I think Python is good enough.” (David, a CS associate professor)

This idea is separately advanced by other interviewees. Per, Kash, “for the sixth graders, definitely you know, it's good to introduce block-based (visual) programming ideas, but for a high school again, my opinion is to introduce Python.” (Kash, a CS associate professor). Daxton holds a similar opinion:

> “They're going to have to know how to do sequence selection iteration, whether it's graphical or not. I think it [block-based visual programming] is good for K-2 to K-5. But once they get to K-6 through 12, I think it should be a text-based programming language.” (Daxton, a CS instructor in higher education)

4.2.2 Code II: Syntax-based Languages: Python, Java, C++

In particular, the specific text-based programming languages that are encouraged included Python, Java, and C++, etc. For example, Kash says, “Python is appropriate for K-12 CS education. In Python, students don't receive too many syntax issues, and they can focus on improving problem-solving

techniques." (Kash, a CS associate professor). Moreover, Kate and Lisa mention that their schools have already taught syntax-based programming language in high school.

> "At the high school, we use Python and Java. We use programming languages and tools that they can utilize. Now we teach an AP Computer Science class. So that does have to be the Java language because that's what the test is on. But those are all very marketable software tools that they can use, whether it's in college or if they decide college is not for them. They can also use in the real world." (Kate, a K-12 high school CS teacher)

> "We also use Python to begin to develop the understanding of what is the language and how do you learn it. By ninth grade, they're doing full-on Python. They can take Java after ninth grade. And so those are both options for continuation" (Lucy, a K-12 middle school CS teacher)

4.2.3 Code III: Specific Programming Language does not Matter

Overall, four interviewees think that the specific programming language is not that important compared to CT skills for problem-solving. K-12 students can learn CT skills without using particular "real" programming language, as indicated in previous cites from Aiden and Kash.

Students can learn any programming language, such as Python, to learn CT skills. Once they master one programming language, the knowledge can be transferred when learning other programming languages. As Daxton, Lucy, and Kate explain below:

> "I think, from what I've seen, there is a lot of emphasis on knowing what language to teach. That is not important. The language is coming today; you learn Python, but two years from now, Python will probably disappear, and other languages will come. So more important is to know one language. Don't focus on learning how to use that language to program things, so computational thinking is more important than the language itself. A language is a tool." (Daxton, a CS associate professor)

> "We try very hard to create a basis of understanding the language, not a specific language, but just what a programming language is and does, and then that way, as languages change, students can still apply the same knowledge to any language." (Lucy, a K-12 CS teacher)

> "We have a beginning and intermediate [class], and then we have the AP [CS] class. So, we have different levels. And once you learn how to do as…if statement, once you know how to do a for loop, you know, you can apply it with any language." (Kate, a K-12 high school CS teacher)

5. Discussion

The primary goal of the current study is to explore the necessity of K-12 CS education, K-12 CS students' core competencies, and programming languages that should be learned in K-12 from CS professors' and teachers' perspectives. The findings of this study reveal that while most interviewees believe that K-12 CS education is necessary, problem-solving skills using computational thinking are the top important competencies in K-12 CS education. In addition, K-12 students should have basic math background and foundational programming skills. Regarding the programming languages, this study found that interviewees suggested starting with a block-based visual programming language and then moving to textual languages, such as Python, Java, C++. However, the specific language was not considered as important as CT and problem-solving skills.

In terms of the importance of computational thinking and problem-solving skills in CS education, the finding of this study aligns with the statements from the prior researchers (Grover & Pea, 2013) that the problem-solving approach is often related to CT skills. Regarding the concepts of CT, some interviewees have a common understanding of using a computational approach, such as abstraction, decomposition, algorithm, and generalization, to solve problems, which aligns with the categories from Selby and Woollard (2013). In addition, CT skills not only could be used in the CS field but also be helpful for other subjects. Researchers explored approaches of integrating CT skills in K-12 through diverse approaches. Sengupta et al. (2013) proposed a theoretical framework for integrating computational thinking in K-12 science education. The framework includes three stages: (1) scientific inquiry, (2) algorithm design, and (3) engineering. Moreover, Yadav et al. (2016) provided suggestions for instructional technologies and training experts for integrating CT into other subjects in K-12. Kwon et al. (2021) implemented CT in primary education using problem-based learning approach and examined the development of CT skills maong students.

Interviewees in this study highlight the importance of math knowledge in CS education. Interviewees consider that math lays the foundations for advanced CS learning, which concurs with argument from Beaubouef (2002) and Konvalina, Wileman, and Stephens (1983). In reality, both CS and math subjects require students to have logical thinking skills. Beaubouef (2002) stated that math is critical in diverse perspectives in CS, including problem-sovling, programming, computer hardware and architecture, CS theory, and softeware engineering. Regarding whether math should be the prerequisite of CS education, especially in K-12 education, no consensus has been achieved yet. Further research can examine the relationship between math and CS education.

The findings of this study also indicate that programming skill is important in K-12 CS education. This finding concurs with the statements from prior researchers, such as Grover and Pea (2013) and Lye and Koh (2014). Programming is an important tool to develop CT skills for problem-solving.

Consequently, deciding on programming languages to be taught in K-12 CS education is essential. This study finds that interviewees hold different perspectives. Some suggest using block-based programming tools such as Scratch for each CT skill. Others suggest teaching some specific widely-used textual programming languages, such as Python, Java, C++, etc., as suggested by TIOBE (2021). Among these diverse opinions, interviewees in this study also suggest letting students start using block-based programming tools in lower grades and gradually introduce textual programming languages in higher grades. Despite that the last perspective compromises the first two opinions, more details need to be explored regarding when and how the transition from visual programming languages to textual programming languages should be put into practice.

Although the interviewees share opinion regarding diverse programming languages, some also emphasize the specific programming languages taught is not that important as long as students can learn CT skills. They highlight that once students learn one programming language to develop their CT skills, they can easily transfer what they have learned to new programming languages. Future research may further examine whether using different programming languages influence their outcome of obtaining CT skills and how to efficiently and effectively transfer between different programming languages.

6. Limitations and Future Research

Some limitations exist in this study. First, this study used the self-reported interview data from volunteers as the data source, which may have bias. Further research can incorporate other data sources, such as policy documents, reports, and observations to confirm or refine findings from this study. Second, the interviewees are from the CS professors and K-12 CS teachers in Michigan State. The generalization of the study findings from this study should be cautious. The status of K-12 CS education in different states is heterogeneous, which may influence their CS professors' and teachers' perspectives. Last, the participants of this study are CS professors and teachers, which leave the key stakeholders of K-12 CS education, students, outside of the conversation. Future research can further explore students' opinions of K-12 CS education.

7. Conclusions

This study's findings indicate the core of CS education includes problem-solving and CT skills, math background, and foundational programming skills. CT is considered an important skill to solve problems, which supports Wing's (2006) definition. Therefore, CT is critical in K-12 CS education. Math may be one of the foundation subjects for CS education. In addition, pre-college experiences in computer programming are important. However, the specific programming language is not the critical

element as long as students master CT and problem-solving skills. K-12 students may start from the visual programming languages and then transfer to textual programming languages. The study findings deepen our understanding of K-12 CS education, which helps educators and policymakers making decisions regarding K-12 CS education.

References

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. Doi: 10.1145/1929887.1929905

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017, June). Learnable programming: Blocks and beyond. In *the Communications of the ACM, 60*(6), 72–80. https://doi.org/10.1145/3015455

Beaubouef, T. (2002). Why computer science students need math. *ACM SIGCSE Bulletin, 34*(4), 57-59. https://doi.org/10.1145/820127.820166

Bernard, H. R., & Ryan, G. W. (2009). *Analyzing qualitative data: Systematic approaches.* SAGE publications.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report,* 68. https://komenskypost.nl/wp-content/uploads/2017/01/jrc104188_computhinkreport.pdf

Bower, M., Wood, L. N., Lai, J. W., Highfield, K., Veal, J., Howe, C., ... & Mason, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education (Online)*, *42*(3), 53-72. https://search.informit.org/doi/abs/10.3316/informit.767807290396583

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology, 3*(2), 77-101.

Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2019). The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages. *Computer Science Education, 29*(1), 23-48. https://doi.org/10.1080/08993408.2018.1547564

Chou, P.-N. (2018). Skill development and knowledge acquisition cultivated by maker education: Evidence from Arduino-based educational robotics. *EURASIA Journal of Mathematics, Science and Technology Education, 14*(10), 1–15. https://doi.org/10.29333/ejmste/93483

Çiftci, S., & Bildiren, A. (2020). The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. *Computer Science Education, 30*(1), 3-21. https://doi.org/10.1080/08993408.2019.1696169

Code.org, CSTA, & ECEP Alliance. (2020). *2020 State of Computer Science Education: Illuminating Disparities*. https://advocacy.code.org/stateofcs

Creswell, J. W., & Clark, V. L. P. (2017). Designing and conducting mixed methods research. Sage publications.

CSTA (n.d.). *Computer science standards.* CSTA. Retrieved from https://www.csteachers.org/page/standards

Dehouck, R. (2016). The maturity of visual programming. http://www.craft.ai/blog/the-maturity-of-visual-programming/

Enbody, R. J., & Punch, W. F. (2010, March). Performance of Python CS1 students in mid-level non-Python CS courses. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 520-523). https://doi.org/10.1145/1734263.1734437

Erlandson, D. A., Harris, E. L., Skipper, B. L., & Allen, S. D. (1993). *Doing naturalistic inquiry: A guide to methods*. Sage.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education (TOCE), 14*(2), 1-18. https://doi.org/10.1145/2602483

Giorgi, A. P., & Giorgi, B. M. (2003). The descriptive phenomenological psychological method. In P. M. Camic, J. E. Rhodes, & L. Yardley (Eds.), *Qualitative research in psychology: Expanding perspectives in methodology and design* (pp. 243–273). American Psychological Association

Gretter, S., & Yadav, A. (2016). Computational thinking and media and information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends, 60*(5), 510–516. https://doi.org/10.1007/s11528-016-0098-4

Grover, S. & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42* (1), 38–43. https://doi.org/10.3102/0013189X12463051

Guest, G. (2012). *Applied thematic analysis*. Sage.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Irish, T., & Kang, N. H. (2018). Connecting classroom science with everyday life: Teachers' attempts and students' insights. *International Journal of Science and Mathematics Education, 16(*7), 1227-1245. Doi: 10.1007/s10763-017-9836-0

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education, 82*, 263-279. https://doi.org/10.1016/j.compedu.2014.11.022

K-12 Computer Science Framework Steering Committee. (2016). K-12 computer science framework. ACM. doi:https://doi.org/10.1145/3079760

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83–137. https://doi.org/10.1145/1089733.1089734

Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). Math proficiency: A key to success for computer science students. *Communications of the ACM, 26*(5), 377-382. https://doi.org/10.1145/69586.358140

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72,* 558-569. https://doi.org/10.1016/j.chb.2017.01.005

Kwon, K., Jeon, M., Guo, M., Yan, G., Kim, J., Ottenbreit-Leftwich, A. T., & Brush, T. A. (2021). Computational thinking practices: Lessons learned from a problem-based curriculum in primary education. *Journal of Research on Technology in Education,* 1-18. https://doi.org/10.1080/15391523.2021.2014372

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry.* Sage.

Lindh, J., & Holgersson, T. (2007). Does lego training stimulate pupils' ability to solve logical problems?. *Computers & Education, 49*(4), 1097-1111. https://doi.org/10.1016/j.compedu.2005.12.008

Lockwood, J., & Mooney, A. (2017). Computational thinking in education: Where does it fit? A systematic literary review. *arXiv preprint arXiv:1703.07659.*

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior, 41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM Sigcse Bulletin, 39*(1), 223-227. https://doi.org/10.1145/1227504.1227388

Ministry of Education. (2014). Computer science: A new curriculum in reform. http://cms.education.gov.il/NR/rdonlyres/0E091CFA-8E73-4C24-96A7-0A6D23E571EA/189697/resource_849760831.pdf

Organisation for Economic Co-operation and Development. (2018). The future of education and skills: Education 2030. *OECD Education Working Papers 23.* https://doi.org/10.1111/j.1440-1827.2012.02814.x

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.

Patton, M. Q. (2015). Qualitative research & evaluation methods: Integrating theory and practice: The definitive text of qualitative inquiry frameworks and options (4th ed.). Thousand Oaks, California: SAGE Publications, Inc.

Saez-Lopez, J., Roman-Gonzalez, M., & Vazquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using Scratch in five schools. *Computers & Education, 97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Schmidt, A. (2016). Increasing Computer Literacy with the BBC micro: bit. *IEEE Pervasive Computing, 15*(2), 5-7. Doi: 10.1109/MPRV.2016.23

Seehorn, D., Pirmann, T., Batista, L., Ryder, D., Sedgwick, V., O'Grady-Cunniff, D., Twarek, B., Moix, D., Bell, J., Blankenship, L., Pollock, L., & Uche, C. (2016). CSTA K-12 Computer Science standards 2016 revised. ACM Press. https://dl.acm.org/doi/pdf/10.1145/2593249?casa_token=zOwW-U2zltcAAAAA:RR8hxGKWuykHfnSlZpB_7z4pMY1oFKSWIm9W8txVT-NE4KLKx4JlagcXvX1w0z84VvEIScrM3xln

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition.* https://eprints.soton.ac.uk/356481/

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*, 351-380. https://doi.org/10.1007/s10639-012-9240-x

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Michigan Department of Education (2020, May). State of Computer Science Education in Michigan. https://www.michigan.gov/documents/mde/State_of_Computer_Science_Education_in_Michigan_Report_709699_7.pdf

The Horizon Report. (2017). K–12 edition. https://www.nmc.org/nmchorizon-k12/

TIOBE index. (2021). https://www.tiobe.com/tiobe-index

Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research, 57*(1), 3-31. https://doi.org/10.1177/0735633117743918

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728. Doi: 10.1007/s10639-015-9412-6

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when?. *Education and Information Technologies, 22*(2), 445-468. Doi: 10.1007/s10639-016-9493-x

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wong, G. K. W., & Cheung, H. Y. (2020). Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments, 28*(4), 438-450. https://doi.org/10.1080/10494820.2018.1534245

World Bank. (2019). Children learning to code: Essential for 21st century human capital.

World Economic Forum. (2015). New vision for education unlocking the potential of technology.

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. *Computer Science Education, 29*(2-3), 177-204. https://doi.org/10.1080/08993408.2019.1565233

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60,* 565-568. https://doi.org/10.1007/s11528-016-0087-7

Yu, P., & Hai, T. (2005). A focus conversation model in consumer research: The incorporation of group facilitation paradigm in in-depth interviews. *Asia Pacific Advances in Consumer Research, 6,* 337–344. https://www.acrwebsite.org/volumes/11931

Zhu, M., & Wang, C. (2023). K-12 Computer Science Teaching Strategies, Challenges, and Teachers' Professional Development Opportunities and Needs. *Computers in the Schools*, 1-22. https://doi.org/10.1080/07380569.2023.2178868

Appendix

Semi-structured Interview Questions – K-12

1. Please briefly introduce yourself.
2. Have you heard of CS standards in Michigan? Does your school make plans to meet the standards?
3. What is your understanding of CS education?
4. Which goals and which competencies are intended in K-12-CS Education?
5. What learning content will be/is delivered in K-12 CS Education?
6. Which programming languages and tools are used in K-12 schools?
7. Which types of assessments were used
8. Who is teaching CS?
9. What are the challenges/concerns about teaching CS in K-12?
10. Who do you seek help from when you encounter challenges?
11. What types of resources, support, or additional teacher training are provided in K-12 CS education?

Semi-structured Interview Questions - Higher education

1. Please briefly introduce yourself.
2. What are the future job opportunities for CS students after they graduate?
3. What goals and competencies are intended in each program/CS education in higher education?
4. What are the common programming languages and tools taught in the CS field in higher education?
5. What are the effective instructional strategies for teaching CS students in higher education? Would you mind giving me an example?
6. What are the challenges that you encountered teaching CS students in higher education?
7. Could you please describe a typical successful learner in CS?
8. Do you think CS in K-12 is necessary? Why?
9. If we plan to offer CS curricula in K-12, what competency do you think students could learn in K-12 to help students learn better in college?
10. What languages or tools should be taught in K-12?

11. What knowledge and skills do you think K-12 CS teachers should have to teach students successfully?

12. If they do not have such knowledge and skills, how do you think we can provide support to K-12 CS teachers?

13. Do you have any suggestions for K-12 CS educators?