International Journal of Computer Science Education in Schools, March 2023, Vol. 6, No. 1

ISSN 2513-8359

Analyzing Computational Thinking Studies in Scratch Programming: A Review of Elementary Education Literature

William H. STEWART[1]
Kwanoo BAEK[2]

[1]Hankuk University of Foreign Studies, Korea
[2]University of Southern California, USA

DOI: 10.21585/ijcses.v6i1.156

Abstract

Computational Thinking (CT) has become popular in recent years and has been recognized as an essential skill for everyone in the digital age. CT literature, however, is at an early stage of development, and there is no consensus among researchers/scholars in the field. To date, many have been unable to concretely explain what CT is, or how to teach and assess this broad skill set. This is particularly evident in different educational contexts and settings such as higher education versus elementary education. The purpose of this cumulative literature review is to examine papers that focus on CT in terms of elementary education, elementary-aged learners, and related issues/considerations in order to provide a better understanding of the CT in an elementary context. An inductive qualitative content analysis was conducted on 58 papers set in elementary school settings about CT from 2010-2020. Five main themes emerged from the review: exploiting tangible blocks in a physical coding environment, integrating *Scratch* into various disciplines through programming, *Scratch* gaming for computational thinking, evaluating computational thinking skills through *Scratch* projects, and teaching and learning methods/factors affecting CT in children. Implications for practice and directions for future research are discussed.

Keywords: Scratch, Computational thinking, Programming, Coding, Elementary Education

1. Introduction

The world has become saturated with digital and computer technology in the 21st century, which in turn has made effective computational tool use a necessary professional and economic skill set (Angeli et al., 2016; Bers, 2010; English, 2016; Miaoulis, 2010; Yadav et al., 2011). Further obfuscating this issue is the simple fact that the logic or principles behind computational tool use, which is known as Computational Thinking (CT), manifest differently in practice depending on the subject matter (Weintrop et al., 2015; Wing, 2017; Yang et al., 2018). In recent years, the discussion on CT has evolved into one not just focused on how CT manifests itself, but also a discussion with a growing call to view CT anew from additional participatory, community, and maker perspectives (Kafai, 2016; Rode et al., 2015), a universal metaphor for reasoning (Henderson et al., 2007), as well as one that envisions CT as an all-encompassing 21st century literacy rather than just a discrete set of skills (diSessa, 2018; Jacob & Warschauer, 2018).

ISSN 2513-8359

1.1 What is Computational Thinking?

While computational tools (e.g., robotics, programming, simulations, computers, music, maker spaces, etc.) are diverse (perhaps seemingly disparate), underlying the effective use of such tools is CT. CT, however, is an umbrella term for a problem-solving process that encompasses numerous sub skills such as abstraction, decomposition, and simulation (Brennan & Resnick, 2012; Henderson et al., 2007; Wing, 2017). While CT is recognized in broad terms, no consensus exists on exactly how it should be defined, or what skills ultimately constitute CT (Barr & Stephenson, 2011; Barr et al., 2011; Bocconi et al., 2016; Grover & Pea, 2013; Kalelioglu et al., 2016; Lammi et al., 2018; Selby & Woollard, 2013; Weintrop et al., 2015, Yadav et al., 2016). Although CT has been brought to the foreground of the discussion in STEM fields and STEM integration over the last 15 years (see Wing, 2017), CT itself is not new; the origins of CT can be traced back much further to the 1970s and the work of Seymour Papert's LOGO programming and procedural thinking (Bers, 2010; Grover & Pea, 2013; Lye & Koh, 2014; Sengupta et al., 2013; Weintrop et al., 2015; Yadav et al., 2011). Nevertheless, while an effort has been made to promote CT in high school by the computer science community, no analogous effort exists in primary or middle schools (Angeli et al., 2016; Jacob & Warschauer, 2018). Barr and Stephenson (2011) suggested that there may be difficulties transferring CT from a development context situated in higher education, to a K-12 context where CT is applied differently. For example, Lee et al. (2011) noted that there are multiple possible domains (e.g., web design, mobile app development, robotics) that can be used to help develop CT processes/skills in students but these domains may not be widely available in K-12 whereas they are far more common in higher education.

Barr and Stephenson (2011) suggested that any definition of CT should be accessible and framed in terms of the classrooms in which it will take place (versus an overly technical definition). Barr et al. (2011) proposed that CT is a unique combination of cognitive skills that enable a novel form of problem-solving. This process is also closely tied to various tools (e.g., computers) and can make aspects of problem-solving (i.e., testing, iteration) far more accessible to learners since they can be automated or enacted at a wide scale. Grover and Pea (2013) discussed an overview of the commonalities of various definitions which included abstraction, systematic information processing, symbol systems and representation, and algorithmic concepts such as flow and control. Selby and Woollard (2013) synthesized a definition of CT based on whether or not there was consensus in the literature regarding a specific skill. They suggested that CT can be defined in terms of thinking abstractly, algorithmically, and in terms of decomposition, generalizations, and evaluations. Rode et al. (2015) included aesthetics, creativity constructing, visualizing, and understanding, whereas Jacob and Warschauer (2018) suggested that CT can be re-defined as a new literacy built on programmatic logic. Voogt et al. (2015) elaborated on this definition, describing computational thinking as a universal attitude and skill set that includes decomposition, abstraction, algorithmic thinking and pattern matching and many more. Consequently, computational thinking is considered as a thought process critical for solving problems in a technology-driven society (Kale et al., 2018). Modern digital technology, which is reliant upon programming and coding, is a domain where CT is thought to be necessary. One popular way to explore CT through programming/coding has been through MIT's visual block-based coding platform, *Scratch.*

1.2 What is Scratch Programming/Coding?

Scratch is a block-based visual coding language created by MIT. Although *Scratch* has primarily been associated with a young learning audience (e.g., Chou, 2020; Rose et al., 2020), it's a user-friendly visual interface where students stack and fit blocks together, rather than write code via complex and technical syntax. This block-building metaphor for programming and coding, however, can encourage CT for beginners in the domain regardless of age (e.g., Dolgopolovas et al., 2015; Korkmaz, 2016; Romero et al., 2017). With Scratch, users can learn the

fundamental principles of programming (e.g., sequences, loops, conditional statements, etc.) by creating their own projects, such as games or animated videos. By providing an accessible learning environment where young learners can think about such concepts and engage in various cognitive processes, Scratch is particularly impactful for developing problem-solving skills (Berikan & Özdemir, 2019; Donley, 2012; Korkmaz, 2016; Topallia & Cagiltayb, 2018). The cognitive benefits of Scratch include the development of logical, analytical, mathematical, and creative thinking skills as a means to approach complex problems in computer programming (Korkmaz, 2016). As a problem-solving process by extension, these skills not only overlap with CT, but are critical in practice such as abstraction, algorithmic thinking, problem solving, pattern recognition, and design-based thinking (Kalelioglu et al., 2016). As these skills are logical and mathematical in nature, they are heavily implemented in programming environments, and Scratch is no exception–it is rather a question of how and why Scratch affects computational thinking.

1.3 Why and How Scratch Affects Computational Thinking?

Scratch facilitates the process of thinking through higher mathematical understanding, problem-solving strategies, and analytical thinking skills (Korkmaz, 2016). Calao et al. (2015) found that an experimental group of 6th grade students (who have received training in Scratch) had shown statistically significant improvement in mathematical knowledge with respect to modeling, reasoning, and problem-solving. Given that the observed skills in the experiment coincided with some of the skills in the domain of computational thinking, it is appropriate to conclude that the improvement of mathematical processes with the assistance of a visual programming environment (Scratch) can also help facilitate the development of computational thinking. Other research, however, has noted no such relationship. Kalelioglu et al. (2016), for example, concluded that the effects of Scratch on the problem solving-skills of 5th grade students did not yield conclusive results. Merely providing a learning environment was insufficient with regards to teaching effectively and observing students' performance gains. Nevertheless, many students in this study enjoyed and wanted to learn more programming since they were able to utilize their creativity to create games. Thus, while there were no discernable effects in terms of statistically significant results, Scratch did have a noticeable effect on the desire of the students to improve on their programming skills. The increase in motivation or desire, however, can still promote computational thinking.

Although computational thinking is not equivalent to programming, computational thinking can be seen as a problem-solving method utilized by a computer scientist or programmer. As Wing (2010) elaborated, computational thinking is "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" (p. 1). In other words, computational thinking is a way of figuring out how to solve a problem and processing information in a method that is concerned with the realm of computer science. However, what this looks like in practice and in certain contexts is poorly described or conflated in the literature (Weintrop et al., 2015; Wing, 2017; Yang et al., 2018) and the impetus for this review Thus, to comprehensively contextualize CT in Scratch for teaching and learning, as well as explore the assessment of CT, in primary school (i.e., K-9) classrooms, we performed a cumulative literature review (see Templier & Paré, 2015) in order to answer the following research questions:

RQ1: How is CT defined/conceptualized in the context of Scratch in elementary education?

RQ2: How is CT taught and assessed in Scratch in the context of elementary education?

2. Method

Since CT literature is at an early stage of development, there is no consensus among researchers/scholars in the field and to date, many have been unable to concretely explain what CT is, or how to teach and assess this broad

ISSN 2513-8359

skill set. Therefore, the manner of this review was cumulative in nature where the goal is to "compile empirical evidence to map bodies of literature and draw overall conclusions regarding particular topics of interest" (Templier & Paré, 2015, p. 120). Further, we also employed a semi-scoping approach in which the review is not only one that accumulates a body of evidence, but one that also can "examine and clarify definitions that are used in the literature" (Munn et al., 2018, p. 3). We also followed various systematic procedures based on several other reviews (see Baek et al., 2020; Hamari et al., 2014; Levy & Ellis, 2006; Nakano & Muniz, 2018; Ramdhani et al., 2014). A summary of the overall process is illustrated in Figure 1. Rather than being linear, it is a recursive approach to examining and synthesizing various literature sources.

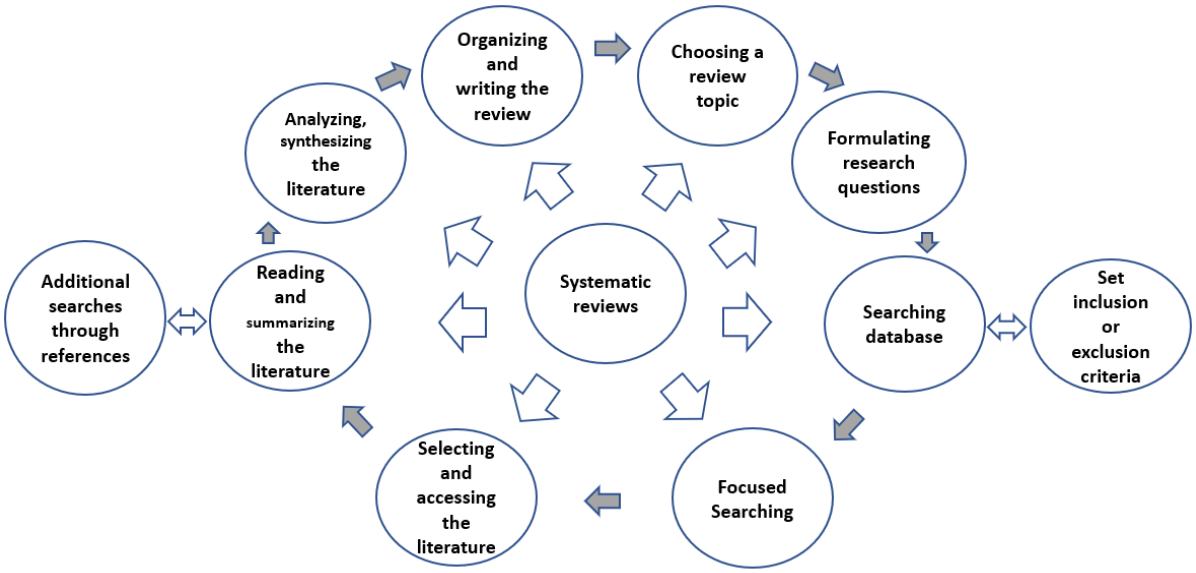

Figure 1. Review Procedure

2.1 Step 1: Search Terms and Databases

Since our investigative target was the use of *Scratch* in elementary educational settings in regards to computational thinking, we used the following keywords, *Scratch*, *computational thinking*, and *education*. We searched Ebscohost, ScienceDirect, Web of Science, Springer, IEEE Digital Library, ACM Digital Library, Google Scholar, and ProQuest for relevant literature displaying these keywords in their titles or abstracts. This initial search yielded 551 articles.

2.2 Step 2: Inclusion and Exclusion Criteria

To focus the scope of this review, we required that articles be written in English, be published from 2010-2020, and be conducted in elementary education. Studies that involved pre- or in-service elementary teacher training were excluded to refine the results to elementary school student performance. This resulted in 125 papers.

ISSN 2513-8359

2.3 Step 3: Assessing the Literature

Papers were screened again by sorting them into two categories: conceptual articles and empirical studies. Conceptual papers discussed the general features of Scratch, provided a theoretical framework or suggested instructional practices for Scratch programming into education. Empirical studies tended to test and justify specific interventions and measure(s) of computational thinking via qualitative, quantitative research, or mixed-methods research designs. This reduced the number of articles to 79.

We further refined the dataset to papers that specifically looked at CT skills (e.g., characteristics and processes, models, assessments, interventions), in addition to including experimental and non-experimental study designs. The research was further scrutinized for the quality of the research designs such as excessive statements or assumptions, tangential CT focus, outcomes other than measures of CT skills, pilot studies, or studies with samples of 10 or less that could not produce valid statistical outcomes. In regards to excluding studies with small sample sizes, scholarship does suggest that the outcomes reported in small studies are more variable than large-study counterparts, ultimately making the results of larger studies more reliable and where the greatest emphasis should be placed (Slavin & Smith, 2009). Therefore, the exclusion of these studies in this review is not to state (or even imply) that such studies are inferior (they are not), rather simply that for the purpose of summarizing evidence for a literature review where broader generalizability of the findings was the goal (Templier & Paré, 2015), the larger studies were preferred for drawing conclusions from (see Slavin & Smith, 2009). Additional assessment of the literature included examining the papers' topical, historical, and methodological relevance, as well as gap analyses. This ultimately produced 58 papers, an overview of which is presented in Figure 2 and 3.

ISSN 2513-8359

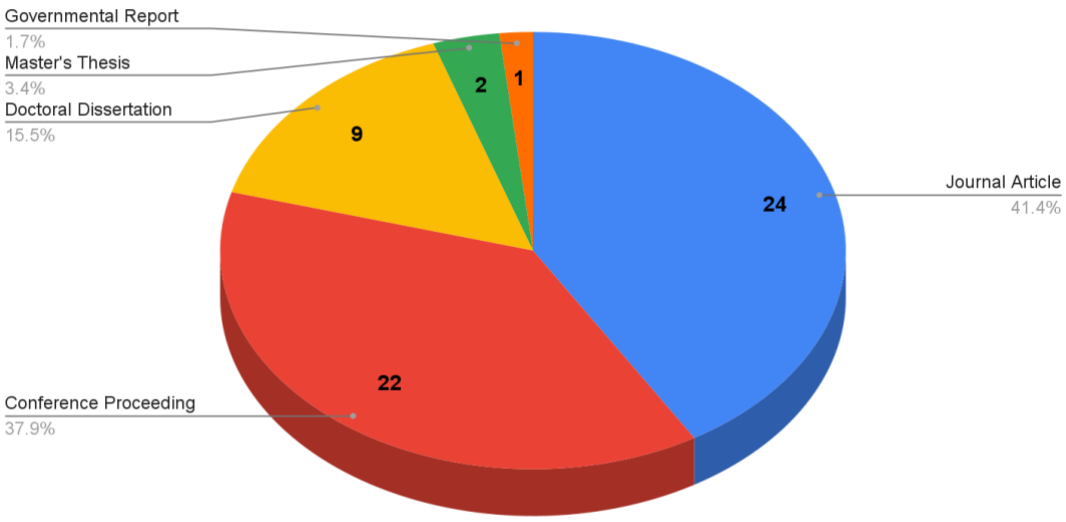

Figure 2. Overview of Review Articles by Type

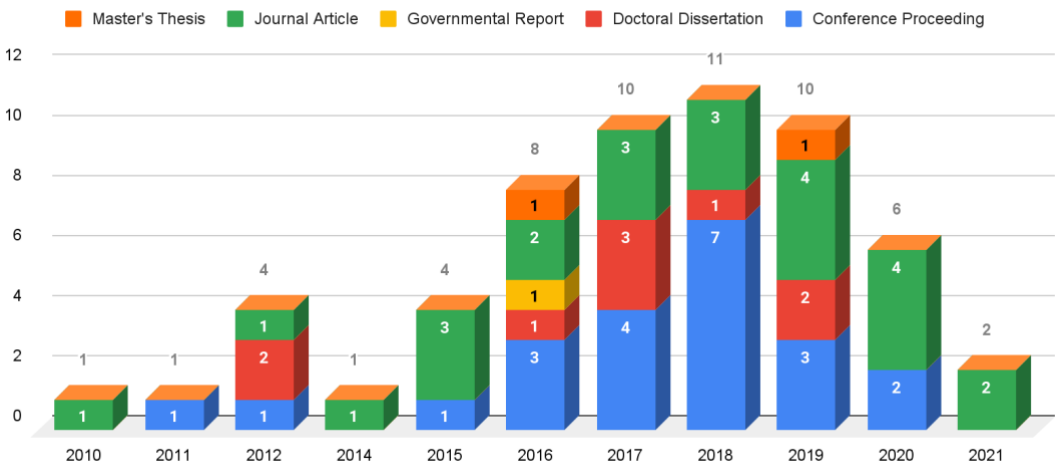

Figure 3. Overview of Review-articles by Year

2.4 Analyzing and Organization

We performed an inductive qualitative content analysis whereby the content discussed in papers (i.e., topics, findings, issues) were assigned keywords or phrases (i.e., codes). These were then aggregated into larger categories where vertical and horizontal relationships appeared (Braun & Clarke, 2006). The researchers discussed areas where thoughts diverged and ultimately, this process produced five core themes which are used to structure the findings section of the review: exploiting tangible blocks in a physical coding environment, integrating *Scratch* into various disciplines through programming, *Scratch* gaming for computational thinking, evaluating

computational thinking skills through *Scratch* projects, and teaching and learning methods/factors affecting CT in children.

3. Results

3.1 Exploiting Blocks in A Coding Environment

Various studies compared Scratch with other programming environments, particularly with their relationship to developing computational thinking skills. Smith (2019) compared Scratch with Cozmo, a robotics-based coding environment While both of these coding and robotics based programming environments shared the same content and instructional features, curriculum with Scratch was often more computer-based whereas the Cozmo curriculum was made of animated emotional-educational robotic activities. Smith (2019) similarly found that students were more engaged when using Cozmo over Scratch, though both programming environments were equivalent in developing CT skills. When Scratch was compared to Lego Mindstorms or C++ environments, Scratch was more effective at developing logical thinking skills (Korkmaz, 2016). Other studies (i.e., da Cruz Alves et al., 2019; Park & Shin, 2019; Quitério Figueiredo, 2017) compared Scratch and App Inventor and found that Scratch projects scored higher when being evaluated on parallelism, synchronization, and flow control whereas App Inventor projects displayed higher scores on user interactivity and data representation. This implies that Scratch, as a tool for coding and enhancing computational thinking skills, has been established, however its efficacy in facilitating other CT skills/knowledge domains is less certain. da Cruz Alves et al. (2019) not only supported this point but cautioned that the evaluation of CT skills is heterogeneous; there is no consensus on exactly what criteria should be used or how to evaluate them. Thus, the comparisons between Scratch, App Inventor, Cozmo, and other coding environments may not be so useful; multiple tools can develop and improve computational thinking (Quitério Figueiredo, 2017; Turchi & Malizia, 2016). Nevertheless, research has shown that visual block-based coding environments do reduce the difficulty of abstract programming concepts and complex syntax by converting them into tangible (metaphorically and physically) and accessible elements for students to manipulate and interact with (Rose et al., 2017). This makes it a valuable tool when integrated into other subject areas and learning environments.

3.2 Integrating Scratch into Various Disciplines Through Programming

Scratch has been shown to be an effective way of developing computational thinking skills when integrated in other disciplines (Moreno-León & Robles, 2016; Olabe et al., 2011; Ruthmann et al., 2010; Scullard et al., 2019). Ruthmann et al. (2010) discussed the potential for developing CT through live musical coding in Scratch by approaching programming as music notation. Olable et al. (2011) noted how Scratch was useful when applied to robotics as the interface between digital code and the real world manifestation of it via a robot, which can provide immediate feedback to learners. Even when Scratch is used by students who are not pursuing conventional computer science or STEM related fields, the use of Scratch can influence the development of computational thinking skills such as abstraction or logical thinking (Harimurti et al., 2018). In short, there are numerous (even unexpected) benefits to integrating Scratch into other disciplines across K-12 such as math, writing, science, or English (Moreno-León & Robles, 2016).

3.2.1 Coding with Scratch for CT enhancement

There is no doubt that coding with Scratch is effective at developing CT. The implementation of Scratch for this has even been refined to include specific sequences of programming projects that progressively challenge learners to think computationally. This kind of curricular structure then requires learners to compose problems, recognize patterns, collect and represent data, and ultimately develop code to solve a particular task or challenge (Swaid &

Suid, 2019). Progress design scenarios such as this help students to learn not only core programming concepts such as sequences, loops, and events, but also computational concepts such as abstracting, modularizing, and debugging (Zhang & Nouri, 2019). Chou (2020) even found that third-grade students improved their CT competence when engaged in weekly Scratch activities. Moreover, parents' active involvement in take-home assignments influenced students' long-term CT competence retention. Fagerlund et al. (2020)'s evaluation of Scratch projects found that students' work indicated CT skills and knowledge in diverse ways, and that nearly all students' projects showed knowledge of patterns, abstraction, collaboration, and logical operators, though less than half of the projects used algorithmic procedure, automation, synchronized parallel scripts, recursive solutions, and boolean logic. Thus, simpler programming and computational thinking tend to be far more prevalent in Scratch projects than more advanced ones; developmental levels need to be considered carefully when implementing Scratch for the purpose of CT development.

Similarly, since computational thinking is a skill related to coding, teaching a subject with coding can also show increased achievement in a subject matter (Calao et al., 2015). Calao et al. (2015) utilized Scratch to see whether it could enhance mathematical understanding among sixth grade students. The results showed that students who received Scratch training gained an increase in the understanding of mathematical processes in modeling process and reality phenomena, reasoning, problem formulation and problem solving, and comparison and execution of procedures and algorithms. Thus, coding in mathematics class is assumed to help develop computational thinking skills. This assumption was later supported by Rodríguez-Martínez et al. (2020) where coding and math performance significantly improved. However, some activities in math could be infused with Scratch coding. For example, Vinayakumar et al. (2018) developed computational exercises facilitating the learning of both fractal geometry and computational thinking through tree drawings, which stimulated learners to think computationally about iteration and size change, leading to the concept of 'parallelism, conditionals, and operators. Nevertheless, while mathematics and programming are perhaps logical and obvious areas to use Scratch in to develop CT, there are other creative ways that Scratch is being used such as storytelling.

3.2.2 Storytelling with Scratch

Storytelling involves both reading and writing, and Scratch has been documented in literature as a novel way to promote computational thinking skills, especially with younger/early grade students (Burke, 2012; Lowe & Brophy, 2019; Smith & Burrow, 2016; Von Gillern, 2017).

Smith and Burrow (2017) analyzed five and seven-year old children's use of CT skills and observed looping actions, debugging, remixing, and expression as the students generated ideas and content for their story, which are all examples of concrete CT skills. Similarly, Lowe and Brophy (2019) concluded that computational thinking seems to be most valuable in young learners when it is grounded in concrete activities such as storytelling. They also argued that students can benefit from spending time in abstract story planning since this bears connection to decomposition and algorithm design. Burke (2012) also noted the potential benefits of storytelling in Scratch which could contribute to enhancing computational thinking skills, arguing that the creative functionality of algorithms accentuates the connection between coding and writing. For this reason, digital stories in Scratch embody the technical and the creative elements of composition (Von Gillern, 2017).

3.3 Scratch Gaming for CT

3.3.1 Playing Scratch games for CT

Rose et al. (2017) designed a computer application, Pirate Plunder, which is a block-based programming game that teaches its players how to use Scratch's coding blocks. It does so by focusing on helping children learn

decomposition and abstraction skills. This is primarily done under the assumption that practicing loops and procedures in Scratch can promote students' abstraction and decomposition skills, thereby improving computational thinking as a result.

Rose et al. (2019) found that Pirate Plunder was effective in using custom blocks, procedures, and clones in Scratch with children aged 10-11. On Scratch abstraction/decomposition and computational thinking tests, students who played Pirate Plunder showed significantly higher scores than other groups' students. In related studies, Rose et al. (2018) ultimately concluded that Scratch game-based learning can increase children's procedural abstraction in Scratch projects as well as their computational thinking skills, in addition to the development of procedural abstraction skills as a result of controllable success conditions and difficulty levels (Rose et al., 2020).

3.3.2 Creating Scratch Games for CT

Computational thinking development for elementary aged children though the creation of games in Scratch is frequently discussed in the literature (e.g., Fadjo, 2012; Serbec et al., 2018; Ternik et al., 2017; Topallia & Cagiltayb, 2018). Nančovska Šerbec et al. (2018) performed a study that compared how primary school students aged eight to 12 thought versus prospective teachers of computer science. They compared two groups' projects and found the differences in the category of logic, synchronization, and parallelism which were explained by the differences in reasoning, complexity, and understanding of simultaneous events. No differences were found in the conceptual categories of flow control, data representation, abstraction, and user interactivity. These results implied, however, that computational thinking skills of elementary students can be promoted with game programming making activities through guided instruction.

Ternik et al. (2017) analyzed a maze game developed by 17 primary students aged between eight and 10 years-old. Their primary goal was to improve students' basic computational thinking skills by making a maze-game in Scratch. After teaching concepts such as sequences, loops, events, and conditionals, the participants developed their own maze-game. According to the neo-Piagetian theory of cognitive development, four students (out of 17) reached a concrete operational stage in their programming abilities. The participants displayed different rates of progress from the sensorimotor to preoperational to concrete operational stages of reasoning within Scratch. Even if they could determine different levels of understanding and abstract thinking, most students reached the developmental level. When using Scratch programming, educators should consider students' level of cognitive development. More specifically in the context of computational thinking skills, instructional methods are also important. For example, step by step practical exercises with the concepts of parallelism and synchronization in tandem with other advanced concepts should be done (Fadjo, 2012; Serbec et al., 2018; Ternik et al., 2017; Topallia & Cagiltayb, 2018).

Fadjo (2012) found that sixth and seventh grade students who received pre-written Scratch code analogues (i.e., visual novel form) created in Scratch when compared to students who were instructed with only code in a virtual environment developed more computational thinking skills and concept knowledge (such as conditional logics and operator patterns). Rose (2019) also tested the assumption that the earlier children begin to develop expertise in computer science, the faster they will be able to develop a holistic understanding of code, including more abstract programming principles like selection, repetition, debugging, variables and procedures. She found that primary school children can understand abstract computer science concepts if the instructional method utilized a structured level progression, ultimately highlighting the importance of synergy between instructional method, learner characteristics, and certain Scratch-based tools. Such tools have been developed over the last 10 years to support educators' assessment of student programming and development of computational thinking skills. While Scratch, the platform, is often the most visible component of programming education, tailored programming tools have been developed to further unlock its learning and educational potential; one example of such a tool is Dr. Scratch.

ISSN 2513-8359

3.4 Evaluating Computational Thinking Skills through Scratch projects

3.4.1 Using Dr. Scratch

Dr. Scratch is a tool that automates analysis of Scratch programs, detecting the presence/absence of certain target characteristics (e.g., conditional statements) of students' work. In addition to identifying these traits, Dr. Scratch then extrapolates and assigns a CT score to projects, thus providing feedback to both educators and learners about the CT skills present in their work (Moreno-León et al., 2015, Moreno-León et al., 2017). What makes Dr. Scratch a powerful tool, however, is the general consensus on its effectiveness (Browning, 2017; Lawanto, 2016; Oluk & Korkmaz, 2016).

Lawanto (2016), for example, concluded that Dr. Scratch was well suited to assess computational skills, which in turn helped teachers understand students' strengths and weaknesses in programming. Browning (2017) conducted a pre/posttest study in which two groups of 5th-6th graders had a treatment group assessed by Dr. Scratch for the presence of CT skills, and where the control group was assessed by other CT tests. They found that the development in students' programming skills in Scratch would relate to similar increases in their computational thinking skills or improvements in their computational thinking levels. Nevertheless, Dr. Scratch, as a tool, is not without its own limitations which both Lawanto (2016) and Browning (2017) noted, chiefly in the area of formative assessment. That is, Dr. Scratch is a summative assessment by nature and not one that provides feedback during the learning process. While this may be an accurate description of the intent behind how Dr. Scratch was designed, other scholars such as Moreno-León et al. (2015) have, in fact, used Dr. Scratch for formative assessment purposes.

For example, Moreno-León et al. (2015) used Dr. Scratch's analysis output on students' projects as a stimulus to encourage students to keep improving their programming skills. They asked students aged 10 to 14 years-old to read Dr. Scratch output (displayed as feedback) and then try to improve their projects using the guidelines and tips offered by the tool. As a result, the students' computational thinking scores increased and students displayed improved coding skills. This was especially noticeable for students with an initial medium (i.e., developing) CT score rather than for students with a high one. Browning (2017), however, took a somewhat different approach to utilizing Dr. Scratch for formative assessment purposes. The difference was in taking certain components of programming into account (i.e., easier versus more difficult tasks or skills). Browning reported that there was a significant increase in abstraction, which they also suggested could be significant in flow control with a larger sample. In terms of other CT skills, there were no significant differences in logic scores, and little variation in data representation scores. Differences in the purpose of assessment aside, other limitations have been noted in the literature.

Brennan and Resnick (2012), for example, discussed the fact that the use/presence of a particular Scratch block (or group of blocks) was not necessarily a strong indicator of any particular mastery or fluency in a CT concept. In other words, students may not really understand why they need to use one block over another, or what a more efficient and/or effective sequence of blocks might be when compared to their own code. Further, the use of a single project (i.e., a single data point) to extrapolate CT scores may be skewed; multiple projects from a single student would need to be evaluated for validity in the assessment results. Moreover, certain key CT skills cannot be measured or assessed by examining the source code of the project alone -for example- debugging code in a project would not necessarily be evident in the final version of the code. Similarly, the creativity involved in remixing an existing project would not necessarily be obvious without comparing/contrasting it with the original.

3.4.2 Other Assessment Tools

In addition to the use of Dr. Scratch, other studies have documented the use of other tools to assess computational thinking skills in other ways. For example, Chou (2020) used a test developed by Strawhacker et al. (2018) to measure CT skills, which focused on debugging and fixing a program, circling the blocks, matching the program, and reverse engineering (reverse engineering is a battery of video-based programming tests). This assessment requires students to view programming questions via video clips and then asks students to provide solutions/answers on a structured answer sheet. Another example is from Saez-Lopez et al. (2016) who developed and used a visual block creative computing test to assess elementary students' CT competence after receiving instruction in Scratch. Zhang and Nouri (2019) examined the computational thinking skills that can be learned by K-9 students through Scratch, based on Brennan and Resnick's (2012) framework. Brennan and Resnick's (2012) framework consists of the three key dimensions of CT: computational concepts, computational practices, and computational perspective, which is one that ultimately Brennan and Resnick's (2012) framework places students as designers of interactive stories, games, and simulations in a holistic assessment approach.

Fagerlund et al. (2020) created a framework using three formative assessment processes to identify areas of CT in Scratch projects: what to teach and learn (i.e., clarifying learning objectives), estimating students' current level of understanding, and analyzing their conceptual encounters with CT. This framework made it possible to perform formative assessments by integrating coding patterns, code constructs, and the extent to which students had conceptual encounters with CT through Scratch projects in elementary classrooms. Fagerlund et al. (2020) is also an example of in-depth insight of students' experiences with diverse areas in CT. It is also one that sets future directions of CT assessment in facilitating students' learning CT through students' Scratch projects when compared with earlier approaches.

3.5 Teaching & Learning Methods /Factors affecting CT of Children

The type of instruction and the context of the instructional materials play a significant role in students' development of CT skills and concept knowledge. In a relatively large study across six schools in the United States (222 K-2 students), Strawhacker et al. (2018) found that educators who demonstrated flexibility in lesson planning and who were responsive to students' needs had a positive effect on students. Further, educators that made a positive impact were also skilled in their use of technology, and were concerned about developing students' independent thinking skills. This highlights the importance of sound pedagogy and teaching in addition to the use of proper tools when developing CT skills and the use of technology. Quality teaching practices aside, more specific pedagogical approaches have been noted in the literature in terms of efficacy.

Fadjo (2012) used a grounded embodied pedagogy called "instructional embodiment" when teaching abstract concepts through the use of direct and imagined embodiments (embodiment refers to physical motion or activity). Fadjo (2012) found statistically significant effects for students who physically embodied (or acted out) predefined instructional materials such as speech and motion blocks. In addition to physical embodiment, imagined embodiment, is another technique found to be useful for teaching and developing CT. With imagined embodiment, students mentally simulate and construct imaginary worlds. The benefits of this approach were students' implementation of more computational structures in their projects. Similarly, using familiar contexts had a significant effect on identifying and implementing the CT skill pattern recognition, although learning CT concepts from an unfamiliar context had a significant positive effect on the implementation of both broadcast/receive couplings and conditional logic and operator patterns. Pérez et al. (2020) used metaphors to teach Scratch programming to children aged nine to 12 and concluded that using metaphors improved knowledge of programming concepts. For example, when teaching loops, they used the metaphor of a hand mixer, and for conditionals, they used an intelligent fridge. One limitation that was noted, however, was that using metaphors

could not be definitively correlated with enhanced students' CT since the CT test was not applicable to students who are younger than 10 (Pérez et al., 2020).

Student collaboration in Scratch was also found to improve CT skills. Hamelburg (2019) found that when sixth graders designed games with help from peers during collaborative coding, their knowledge of CT improved. While some students experience difficulty in collaborating, students often assisted their peers by making them feel more comfortable during the challenge. The results of this study corroborate the findings of Chowdhury (2017) where collaborative coding was similarly found to improve computational thinking skills. Other studies, however, did not find any positive effect from collaboration on Scratch programming or on CT skills from (Donley, 2012). Marcelino et al. (2018) also obtained similar results as Donley (2012) with adult (teacher trainees) learners of Scratch.

4. Discussion

4.1 A Variety of CT Definitions

There is a general consensus that programming skills are closely related to CT skills, but more importantly the distinction that programming skills are a subset of CT. The literature consistently describes that CT includes all concepts that computer scientists use to solve computational skills. While this conceptual hierarchy is clear, numerous issues arise when considering which concepts are more appropriate to learn at the elementary school level. For example, pedagogical approaches, content/skill scope and sequence, etc. For this reason, some studies emphasize CT concepts differently in the context of elementary school children. Thus, in regards to our first research question regarding how CT is defined and/or conceptualized, we had difficulty in interpreting previous studies and making generalizations from their results given the extant variety. Thus, the interpretation of results from CT studies needs careful and well-reasoned considerations as the variables being manipulated and/or outcomes being measured/assessed are not necessarily the same. To this point, Rose et al. (2017) proposed that future research in this area should focus on the individual concepts involved in computational thinking to get a deeper understanding of CT for elementary students.

4.2 Assessment & Evaluation

To assess CT skills in Scratch programming, research to date has predominantly relied on code analysis of students' projects. While this approach can provide CT competency feedback in the form of a score (Alves et al., 2019), automated calculation in the CT assessment lacks context that observations or interviews can provide. In other words, more comprehensive evaluation and assessment strategies are needed. Further, automated and performance-based approaches also lack explicit suggestions or tips on how to improve code such as in its efficiency or complexity. In Scratch assessment, the use of formative assessment, such as students' explaining parts of their projects and finding mistakes in their code, would be beneficial for developing their CT skills (Ternik et al., 2017). At present, assessment is varied; we suspect that the diversity in assessment and evaluation is strongly correlated with the variety of operational definitions of CT. While this pedagogical/curricular relationship is not novel or unique to CT, it does highlight an ongoing challenge and obstacle for practitioners and researchers. Some variables that are connected to how CT is taught and consequently assessed A number of variables were noted in the literature that affected Scratch performance.

4.3 Variables Affecting CT

Current key CT variables ranged from pedagogical approaches (see Strawhacker et al., 2018), learner gender (e.g., Chou, 2018), previous programming experience, and math skills. However, findings from Longi (2016) regarding

college students' competence in learning programming may provide insight regarding the potential factors influencing CT competence for elementary school aged learners. In Longi's (2016) systematic literature review, two major factors surfaced: namely students' background information and psychological characteristics in terms of performance in programming courses. These two variables may be related to age or a proxy for age, thus, there is a gap in the literature in terms of sampling that warrants additional research with elementary aged learners. In terms of gender, some studies (i.e., Oluk & Korkmaz, 2016) have assessed both Scratch programming (via Dr. Scratch) and CS skills (via the Computational Thinking Levels Scale which includes 5 factors: a) creativity, b) problem solving, c) algorithmic thinking, d) collaboration, and e) critical thinking). Dr Scratch has a gender parameter in its evaluation process however results showed no difference in gender or time online, although there was a significant relationship between programming skill and computational thinking. Further research regarding the effects of gender on CT and/or programming skills, especially across various related knowledge domains, is warranted.

5. Conclusion

CT is developing practice, field, and area of study that has emerged over the last 20 years. In addition to the inevitable growing pains of a nascent topic, discussion and debate has also emerged regarding the principles that ultimately constitute CT and how to develop CT skills. Similarly, how to assess or measure CT has evolved over time from more rudimentary and limited constructs to more holistic approaches. Equally important, however, is the ongoing inquiry into how CT skill teaching, learning, and development manifests for learners of different ages and in different contexts. The findings from this review highlight how exploiting tangible blocks in a physical coding environment can be particularly beneficial for young/novice learners, and that integrating Scratch into various disciplines through programming (not just programming alone) demonstrates increased learning gains and CT skill development. Additionally, more pedagogically sound ways of teaching CT skills and Scratch have emerged with demonstrated learning effectiveness, and in tandem how CT and programming skills are assessed is starting to evolve in more holistic and sophisticated ways than in years prior. Similarly, more discrete variables affecting CT skill development in children have been identified and are starting to be researched. However, given the emergent or nascent character of CT as a field of practice and inquiry, ongoing research is warranted in several areas.

Current literature has only begun looking into how students interact with the Scratch interface. While the visual block-building metaphor has been effective since Scratch's inception for young learners when compared to learning syntax, some research has shown that younger learners interact differently with programming interfaces such as Scratch, Scratch Jr., and Lightbot. While research findings have generally been in line with the pedagogical underpinnings of Scratch and Scratch Jr., future research can investigate the different approaches that students, particularly young ones, take to programming through visual interfaces.

In this review we found only limited literature that looked at how elementary school teachers are learning about Scratch and CT, and particularly the most effective ways of teaching it as subject matter as well as a critical thinking skill. There is burgeoning discussion about teaching methods that moved beyond just the pedagogical foundations of Scratch as a platform (i.e., constructivism) and into best practices with the platform (e.g., teaching metaphors, generative strategies [instructional embodiment], robust assessment methods, etc.). As current literature is only beginning to investigate and describe more deliberate uses of formative assessment with Scratch and CT, in addition to holistic approaches, research is warranted in this area in general, and specifically with elementary aged learners.

References

Alves, N. D. C., Von Wangenheim, C. G., & Hauck, J. C. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education, 18*, 17–39. http://doi.org/10.15388/infedu.2019.02

Angeli, C., Voogt, J., Fluck, A. E., Webb, M., Cox, M. J., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework - Implications for teacher knowledge. *Educational Technology & Society*, *19*, 47–57. https://www.jstor.org/stable/pdf/jeductechsoci.19.3.47.pdf

Baek, Y., Min, E., & Yun, S. (2020) Mining educational implications of Minecraft. *Computers in the Schools*, *37*, 1–16. http://doi.org/10.1080/07380569.2020.1719802

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*, 20–23. https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads*, *2*, 48–13. http://doi.org/10.1145/1929887.1929905

Berikan, B., & Özdemir, S. (2019). Investigating "problem-solving with datasets" as an implementation of computational thinking: A literature review. *Journal of Educational Computing Research*, *58*, 502–534. https://doi.org/10.1177/0735633119845694

Bers, M. U. (2010). The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice*, *12*, 1–20. https://files.eric.ed.gov/fulltext/EJ910910.pdf

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implication for policy and practice*. Joint Research Center (JRC) Science for Policy Report. https://ec.europa.eu/jrc

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*, 77–101. https://doi.org/10.1191/1478088706qp063oa

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association* (pp. 1–25). AERA. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Browning, S. F. (2017). *Using Dr. Scratch as a formative feedback tool to assess computational thinking*. [Master's thesis, Brigham Young University]. https://scholarsarchive.byu.edu/etd/6659

Burke, W. Q. (2012). *Coding & composition: Youth storytelling with Scratch programming* (Publication No. 3510989) [Doctoral dissertation, University of Pennsylvania]. ProQuest Dissertations Publishing.

Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with Scratch. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & É. Lavoué (Eds.) *Design for teaching and learning in a networked world* (pp. 17–27). Springer International Publishing. http://doi.org/10.1007/978-3-319-24258-3 2

Chou, P.-N. (2018). Smart technology for sustainable curriculum: Using drone to support young students' learning. *Sustainability*, *10*, 3819. https://doi.org/10.3390/su10103819

Chou, P.-N. (2020). Using ScratchJr to foster young children's computational thinking competence: A case study in a third-grade computer class. *Journal of Educational Computing Research, 58*, 570–595. https://doi.org/10.1177/0735633119872908

Chowdhury, B. T. (2017). *Collaboratively learning computational thinking*. Unpublished doctoral dissertation, of Virginia Polytechnic Institute and State University, USA.

da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education, 18*, 17–39. https://doi.org/10.15388/infedu.2019.02

diSessa, A. A. (2018). Computational literacy and "The Big Picture" concerning computers in mathematics education. *Mathematical thinking and learning, 20*, 3–31. https://doi.org/10.1080/10986065.2018.1403544

Dolgopolovas, V., Jevsikova, T., Savulionienė, L., & Dagienė, V. (2015). On evaluation of computational thinking of software engineering novice students. In *Proceedings of the IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations* (pp. 90–99). https://core.ac.uk/download/pdf/42583209.pdf#page=98

Donley, K. S. (2012). *Coding in the curriculum: learning computational practices and concepts, creative problem solving skills, and academic content in ten to fourteen-year-old children* (Publication No. 10842428) [Doctoral dissertation, Temple University]. ProQuest Dissertations Publishing.

English, L. D. (2016). STEM education K-12: Perspectives on integration. *International Journal of STEM Education, 3*, 1–8. http://doi.org/10.1186/s40594-016-0036-1

Fadjo, C. L. (2012). *Developing computational thinking through grounded embodied cognition* (Publication No. 3506300) [Doctoral dissertation, Columbia University]. ProQuest Dissertations Publishing.

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Assessing 4th grade students' computational thinking through scratch programming projects. *Informatics in Education, 19*, 611–640. https://doi.org/10.15388/infedu.2020.27

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

Hamari, J., Koivisto, J., & Sarsa, H. (2014, January). Does gamification work? – A litera-

ture review of empirical studies on gamification. *Proceedings of the 47th Hawaii International Conference on System Sciences* (pp. 3025–3034). Waikoloa, Hawaii, USA. http://doi.org/10.1109/HIS.2014.377

Hamelburg, N. (2019). *Coding, collaboration, and computational thinking* (Publication No. 10183306) [Master's thesis, Hofstra University]. ProQuest Dissertations Publishing.

Harimurti, R., Qoiriah, A., Ekohariadi, E., & Munoto, M. (2018, July). Implementation of computational thinking concepts in ICT learning using Scratch programming. In *International Conference on Indonesian Technical Vocational Education and Association (APTEKINDO 2018)* (pp. 105–109). Atlantis Press. https://doi.org/10.2991/aptekindo-18.2018.23

Henderson, P. B., Cortina, T. J., & Wing, J. M. (2007). Computational thinking. In *Proceedings of the 38th SIGCSE Technical Symposium* (pp. 195–3). ACM Press. http://doi.org/10.1145/1227310.1227378

Jacob, S. R., Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration*, *1*, 1–21. http://doi.org/10.26716/jcsi.2018.01.1.1

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, *59*, 26–27. http://doi.org/10.1145/2955114

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends*, *62*, 574–584. https://doi.org/10.1007/s11528-018-0290-9

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking Based on a systematic research review. *Baltic Journal of Modern Computing*, *4*, 583–596. http://acikerisim.baskent.edu.tr/handle/11727/3831

Korkmaz, Ö. (2016). The effect of Scratch- and Lego Mindstorms Ev3-based programming activities in academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *Malaysian Online Journal of Educational Sciences*, *4*, 73–88. https://ajap.um.edu.my/index.php/MOJES/article/download/12658/8149

Lammi, M., Denson, C., & Asunda, P. (2018). Search and review of the literature on engineering design challenges in secondary school settings. *Journal of Pre-College Engineering Education Research*, *8*, 1–19. http://doi.org/10.7771/2157-9288.1172

Lawanto, K. N. (2016). *Exploring trends in middle school students' computational thinking in the online scratch community: A pilot study* (Publication No. 10183306) [Master's thesis, Utah State University]. ProQuest Dissertations Publishing.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*, 32–7. http://doi.org/10.1145/1929887.1929902

Levy, Y., & Ellis, T. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science: The International Journal of an Emerging Transdiscipline*, *9*, 181–212. http://doi.org/10.28945/479

Longi, K. (2016). *Exploring factors that affect performance on introductory programming courses* (Unpublished master's thesis). Department of Computer Science, University of Helsinki, Finland.

Lowe, T. A., & Brophy, S. P. (2019, June). Identifying computational thinking in storytelling literacy activities with Scratch Jr. In *2019 ASEE Annual Conference & Exposition*. https://peer.asee.org/32913

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. http://doi.org/10.1016/j.chb.2014.09.012

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning computational thinking and Scratch at distance. *Computers in Human Behavior*, *80*, 470–477. https://doi.org/10.1016/j.chb.2017.09.025

Miaoulis, I. (2010). K-12 engineering – The missing core discipline. In *Holistic engineering education* (pp. 37–51). Springer New York. http://doi.org/10.1007/978-1-4419-1393-7_4

Moreno-León, J., & Robles, G. (2016, April). Code to learn with Scratch? A systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 150–156). IEEE. https://doi.org/10.1109/EDUCON.2016.7474546

Moreno-León, J., Robles, G., & Román-González. (2015). Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia, 15*, 1–23 https://www.um.es/ead/red/46/moreno_robles.pdf

Moreno-León, J., Robles, G., & Román-González, M. (2017). Towards data-driven learning paths to develop computational thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing*, *8*, 193–205. https://doi.org/10.1109/TETC.2017.2734818

Munn, Z., Peters, M. D., Stern, C., Tufanaru, C., McArthur, A., & Aromataris, E. (2018). Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach. *BMC Medical Research Methodology*, 18, 1–7. https://doi.org/10.1186/s12874-018-0611-x

Nakano, D., & Muniz, J. Jr., (2018). Writing the literature review for an empirical paper. *Production, 28*, e20170086. http://doi.org/10.1590/0103-6513.20170086

Nančovska Šerbec, I., Cerar, Š., & Žerovnik, A. (2018). Developing computational thinking through games in Scratch. *XI Национална конференция Образованието и изследванията в информационното общество 2018* [*XI National Conference "Education and Research in the Information Society 2018*]. http://pefprints.pef.uni-lj.si/5141/1/Serbec_Developing.pdf

Olabe, M., Basogain, X., Maíz, I., & Castano, C. H. (2011). Programming and robotics with Scratch in primary education. In A. Méndez-Vilas (Ed.), *Education in a technological world: Communicating current and emerging research and technological efforts* (pp. 355-363). Formatex Research Centre.

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' Scratch skills with their computational thinking skills in terms of different variables. *Online Submission*, *8*, 1–7. http://doi.org/10.5815/ijmecs.2016.11.01

Park, Y., & Shin, Y. (2019). Comparing the effectiveness of Scratch and App Inventor with regard to learning computational thinking concepts. *Electronics*, *8*, 1269. https://doi.org/10.3390/electronics8111269

Pérez, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, *105*, 105849. https://doi.org/10.1016/j.chb.2018.12.027

Quitério Figueiredo, J. A. Q. (2017). How to improve computational thinking: A case study. *Education in the Knowledge Society*, *18*, 35–51. https://doi.org/10.14201/eks20171843551

Ramdhani, A., Ramdhani, M. A., & Amin, A. S. (2014). Writing a literature review research paper: A step-by-step approach. *International Journal of Basic and Applied Science*, *3*,47–56. http://digilib.uinsgd.ac.id/5129/1/08IJBAS%283%29%281%29.pdf

Rode, J. A., Booker, J., Marshall, A., Weibert, A., Aal, K., Rekowski, von, T., et al. (2015). From computational thinking to computational making. In *2015 ACM International Joint Conference* (pp. 401–402). ACM Press. http://doi.org/10.1145/2800835.2800926

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020) Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, *28*, 316–327. https://doi.org/10.1080/10494820.2019.1612448

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*, 1–15. https://doi.org/10.1186/s41239-017-0080-z

Rose, S. P. (2019). *Developing children's computational thinking using programming games* (Publication No. 27771989) [Doctoral dissertation, Sheffield Hallam University]. ProQuest Dissertations Publishing.

Rose, S. P., Habgood, M. P. J., & Jay, T. (2017). An exploration of the role of visual programming tools in the development of young children's computational thinking. *The Electronic Journal of e-Learning*, *15*, 297–309. https://doi.org/10.34190/ejel.15.4.2368

Rose, S., Habgood, J., & Jay, T. (2018). Pirate Plunder: Game-based computational thinking using Scratch blocks. In *Proceedings of the 12th European Conference on Games Based Learning* (pp. 556–564). Academic Conferences and Publishing International Limited. https://core.ac.uk/download/pdf/160276026.pdf

Rose, S. P., Habgood, M. J., & Jay, T. (2019, May). Using Pirate Plunder to develop children's abstraction skills in Scratch. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–6). https://core.ac.uk/download/pdf/189171289.pdf

Rose, S., Habgood, J., & Jay, T. (2020). Designing a programming game to improve children's procedural abstraction skills in Scratch. *Journal of Educational Computing Researc*h, *58*, 1372–1411. https://journals.sagepub.com/doi/pdf/10.1177/0735633120932871

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, C. (2010, March). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 351–355). https://dl.acm.org/doi/abs/10.1145/1734263.1734384

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

Scullard, S., Tsibolane, P., & Garbutt, M. (2019). The role of Scratch visual programming in the development of computational thinking of non-is majors. In *Proceedings 2019 Pacific Asia Conference on Information Systems (PACIS)* (pp. 79*)*. https://aisel.aisnet.org/pacis2019/79

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. Corwin Press. http://doi.org/10.4135/9781506313214

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. http://doi.org/10.1007/s10639-012-9240-x

Serbec, I. N., Cerar, Š., & Zerovnik, A. (2018). Developing computational thinking through games in scratch. In *Proceedings at 11th National Conference with International Participation, Education and Research in the Information Society* (pp. 21–30). Plovdiv, Bulgaria

Slavin, R., & Smith, D. (2009). The relationship between sample sizes and effect sizes in systematic reviews in education. *Educational Evaluation and Policy Analysis*, *31*, 500–506. https://doi.org/10.3102/0162373709352369

Smith, S. M. (2019). *A comparison of computer-based and robotic programming instruction:Impact of scratch versus cozmo on middle school students' computational thinking, spatial skills, competency beliefs, and engagement* (Publication No. 27602977) [Doctoral dissertation, Kent State University]. ProQuest Dissertations Publishing.

Smith, S., & Burrow, L. E. (2016). Programming multimedia stories in Scratch to integrate computational thinking and writing with elementary students. *Journal of Mathematics Education, 9*, 119–131. https://educationforatoz.com/images/2016_Commentary_6.pdf

Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teacher's rules: Exploring the impact of teaching styles on young children's programming knowledge in Scratch Jr..*International Journal of Technology and Design Education*, *28*, 347–376. https://doi.org/10.1007/s10798-017-9400-9

Swaid, S., & Suid, T. (2019, December). Computational thinking education: Who let the dog out?. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 788–792). IEEE. http://doi.org/10.1109/CSCI49370.2019.00150

Templier, M., & Paré, G. (2015). A framework for guiding and evaluating literature reviews. *Communications of the Association for Information Systems*, *37*, 112–137. https://doi.org/10.17705/1CAIS.03706

Ternik, Ž., Koron, A., Koron, T., & Šerbec, I. N. (2017). Learning programming concepts through maze game in Scratch. In *Proceedings at 11th European Conference on Games Based Learning* (pp. 661–670). Academic Conferences International Limited.

Topallia, D., & Cagiltayb, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers and Education*, *120*, 64–74. https://doi.org/10.1016/j.compedu.2018.01.011

Turchi, T., & Malizia, A. (2016). Fostering computational thinking skills with a tangible blocks programming environment. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 232–233). IEEE. https://doi.org/10.1109/VLHCC.2016.7739692

Vinayakumar, R., Soman, K. P., & Menon, P. (2018, July). Fractal geometry: Enhancing computational thinking with MIT Scratch. In *2018 9th International Conference on Computing, Communication and Networking Technologies* (ICCCNT) (pp. 1-6). IEEE. https://doi.org/10.1109/ICCCNT.2018.8494172

Von Gillern, S. (2017). *Young children, computer coding, and story creation: An examination of first- and second-grade children's multimodal stories and literacy practices when engaged with a multimedia coding application* (Publication No. 10269304) [Doctoral dissertation, Iowa State University]. ProQuest Dissertations Publishing.

Voogt, J., Fisser, P., & Good, J. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*, 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. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*, 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2010, November 17). Computational thinking: What and why. *The Link*. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wing, J. M. (2017). Computational thinking's influence on research and education for all.*Italian Journal of Educational Technology*, *25*, 7–14. https://doi.org/10.17471/2499-4324/922

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. http://doi.org/10.1007/s11528-016-0087-7

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

Yang, D., Swanson, S., Chittoori, B., & Baek, Y. (2018). Integrating computational thinking in stem education through project-based learning. In *Proceedings of the 5th STEM in Education Conference*. ASEE. https://par.nsf.gov/biblio/10106769

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607