

# Mapping Computational Thinking Skills Through Digital Games Co-Creation Activity Amongst Malaysian Sub-urban Children

Journal of Educational Computing Research

2022, Vol. 0(0) 1–35

© The Author(s) 2022



Article reuse guidelines:

[sagepub.com/journals-permissions](https://sagepub.com/journals-permissions)

DOI: 10.1177/07356331221121106

[journals.sagepub.com/home/jec](https://journals.sagepub.com/home/jec)



Mohd Kamal Othman<sup>1</sup> , Syazni Jazlan<sup>1</sup>, Fatin Afiqah Yamin<sup>1</sup>, Shaziti Aman<sup>1</sup>, Fitri Suraya Mohamad<sup>1</sup> , Nurfarahani Norman Anuar<sup>1</sup>, Abdulrazak Yahya Saleh<sup>1</sup>, and Ahmad Azaini Abdul Manaf<sup>2</sup>

## Abstract

This study investigates how digital game co-creation promotes Computational Thinking (CT) skills among children in sub-urban primary schools. Understanding how CT skills can be fostered in learning programming concepts through co-creating digital games is crucial to determine instructional strategies that match the young students' interests and capacities. The empirical study has successfully produced a new checklist that can be used as a tool to describe the learning of CT skills when children co-create digital games. The checklist consists of 10 core CT skills: abstraction, decomposition, algorithmic thinking, generalisation, representation, socialisation, code literacy, automation, coordination, and debugging. Thirty-six 10–12 year-olds from sub-urban primary schools in Borneo participated in creating games in three separate eight-hour sessions. In addition, one pilot session with five participants was conducted. The game co-creation process was recorded to identify and determine how these young, inexperienced, untrained young learners collaborated while using CT skills. Analysis of their narratives while co-creating digital games revealed a pattern of using CT while

<sup>1</sup>Faculty of Cognitive Sciences and Human Development, Universiti Malaysia Sarawak, Sarawak, Malaysia

<sup>2</sup>Faculty of Creative Technology and Heritage, Universiti Malaysia Kelantan, Kelantan, Malaysia

## Corresponding Author:

Mohd Kamal Othman, Faculty of Cognitive Sciences and Human Development, Universiti Malaysia Sarawak, FCSHD, Sarawak 94300, Malaysia.

Email: [omkamal@unimas.my](mailto:omkamal@unimas.my)

developing the games. Although none of the groups demonstrated the use of all ten CTs, conclusively, all ten components of the CT were visibly present in their co-created digital games.

### **Keywords**

Digital games development, co-creation, computational thinking, children, sub-urban

## **Introduction**

The ability to think computationally is a necessary component of education for the twenty-first century, and according to [Tekdal \(2021\)](#), the field's research has expanded rapidly in recent years. Higher-order thinking skills are being pushed in Malaysian classrooms to keep up with the demands of the modern learning environment. The primary school curriculum in Malaysia promotes the use of CT and other forms of critical and creative thinking. In accordance with the aspirations of the Malaysian National Blueprint for Education 2015–2025, the current curriculum for all schools under the Ministry of Education offers CT skills through a subject called Technology Across the Curriculum.

CT is a problem-solving skill set traditionally associated with the Computer Science field. It entails conceptualising, theorising and designing systems that overlap with the science of thought, an essential computing component ([Wing, 2006](#)). [Grover and Pea \(2013\)](#) provided a thorough review of CT and a good starting point for understanding the use of CT in K-12 education. CT is a core component of problem-solving that focuses on the cognitive process, whilst code literacy means that students can read and write programming language in learning a programming language ([Vee, 2013](#)). As a result, code literacy will significantly impact digital literacy. Digital literacy includes mastering basic computing skills to create multimodal texts and utilising technology to interact with the world around them ([Blummer, 2008](#)).

Previous studies have established the need to integrate CT skills into the curriculum (e.g., [Balanskat & Engelhardt, 2014](#); [Brown et al., 2013](#); [García-Peñalvo et al., 2016](#); [Lee et al., 2011](#); [Ung et al., 2022](#)) and its perceived impact on education ([García-Peñalvo & Mendes, 2018](#)). [Hambrusch et al. \(2009\)](#) described the importance of CT in K-12 STEM (Science, Technology, Engineering, and Mathematics) education. Numerous studies have emphasised using programming languages to promote CT (e.g., [Tedre, 2017](#); [Lye & Koh, 2014](#); [Kelleher & Pausch, 2005](#)). It is vital to introduce CT skills to learners before learning about programming concepts ([Qualls & Sherrell, 2010](#)), and programming course robots have significantly impacted student CT and creativity in elementary school ([Noh & Lee, 2020](#)). However, studies have consistently found it difficult to ascertain the skills and abilities needed to develop CT skills among learners (e.g., [de Araujo et al., 2016](#); [Tang et al., 2020](#)) or teach CT ([Guzdial, 2008](#)). Recently, [Jiang et al. \(2021\)](#) describe the importance of student participation in the

programming process, which involves social interaction amongst the children and focuses not only on developing CT skills. In addition, [Anuar et al. \(2020\)](#) posited that CT can be introduced in a playful approach to young learners living in remote rural and found out that boys performed better at drawing and abstraction skills whilst the girls did better at recognising patterns and colours.

Currently, there are two leading Game-Based Learning approaches to facilitate the development of CT skills and programming courses, such as “learning through the exercise of designing games” and “learning through gameplay” ([Kazimoglu et al., 2012](#)). This current study employs the first method and focuses on the specific task of developing digital games through co-creation. It builds on an earlier observation about how digital game development is highly motivational and practical in getting students engaged with programming concepts ([Wilson et al., 2011](#)). Moreover, when put into a digital game designer role, young learners will learn to program and develop technology literacy ([Resnick & Silverman, 2005](#)).

Creativity is the ability to construct and create ideas, objects, and inventions that carry value, originality, and effectiveness ([Runco & Jaeger, 2012](#)). [Romero et al. \(2018\)](#) defined co-creation as a collaborative effort to undertake a complex task that requires critical and creative thinking skills, which implies working in a team to develop ideas and solutions ([Romero et al., 2018](#)). Furthermore, [Arnab et al. \(2019\)](#) found how co-creativity facilitates an engaging learning process amongst undergraduates in the UK and how co-creativity experiences enabled the discovery of empathy, purpose, meaning, art, creativity, and teamwork.

### *Motivation/Rationale*

Building on our understanding of previous work on CT and Creativity, we want to investigate if the co-creation process positively affects young learners developing CT skills. The objective of the study is two-fold; (1) to produce a CT skills checklist for learning programming concepts using document analysis, and (2) to validate the produced CT skills by determining the pattern of utilising CT amongst young sub-urban primary school students as they co-create digital games using Scratch™.

Many previous studies have focused on CT skills for various contexts and various CT Skills models/frameworks/definitions/concepts were introduced. For example, [Zhang and Nouri \(2019\)](#) posited that CT skills definition varies between countries, curricula and literature, whilst recently, [Ezeamuzie and Leung \(2022\)](#) in their systematic literature review explained why CT skills have been operationalised differently in the literature, especially empirical studies whereby many of them were aligned with computer science concept and practices. Furthermore, previous studies lack a standard definition of CT or agree on the standard definition ([Grover & Pea, 2013](#)). Consequently, it is difficult for future researchers to choose which definition, concept, model, or construct to use.

The CT skills for the developed digital games were also examined using methodologies and tools that were already in use, such as DrScratch ([Moreno-León & Robles, 2015a, 2015b](#)),

Scratch Analysis Tool (SAT) (Boe et al., 2013), Hairball (Wolz et al., 2011), and Scrape (Denning, 2003). However, following a thorough literature review, we found that the limits of the CT skills utilised in their research restricted us from using these tools. For instance, only seven CT concepts—abstraction and issue decomposition, parallelism, logical reasoning, synchronisation, flow control, user activity, and data representation—are used by Dr Scratch to analyse the scripts. By creating a thorough checklist that can be used to evaluate CT skills in the co-created digital games, this study intends to close these gaps.

Furthermore, this study presents a more comprehensive checklist to measure CT for learning programming concepts and further validates them with young sub-urban children who willingly engage in co-creation by analysing activities and digital game artefacts created using Scratch™. It is crucial to observe the phenomenon, as these participants come from limited technological backgrounds at home and school. Given their low socio-economic backgrounds, they would not participate in a commonly assumed conducive nurturing environment to cultivate interest or skills in computer programming at home or in their schools. In this study, we also intend to document the viability of integrating ICT competencies through co-design by creating opportunities to construct digital games using Scratch™ for children from underprivileged families (particularly in sub-urban areas) who otherwise would not participate in creative technology. Scratch™ was selected for this study because it represents a simplified visual approach to programming and has been notably reported to motivate young people to learn how to program (e.g., Brennan & Resnick, 2013; Maloney et al., 2010; Meerbaum-Salant et al., 2013; Resnick et al., 2009; Wolz et al., 2008).

The rest of the paper is organised as follows. The literature on CT and digital games is reviewed in the part that follows by highlighting the various tools, constructs, definitions, and interpretations. The methodology section focuses on how the instrument was created and how the experiment was done to validate it. The results of the experiment to verify the instrument are reported in the Results section (digital games co-creation activities). The study's goal, which addressed the study's gap, is revisited in the discussion part, followed by a section on future research and conclusions.

## Literature Review

*Digital Games Development.* The development and availability of numerous student-friendly applications, digital game development are becoming more reachable for students (Resnick et al., 2003). Denner et al. (2012) suggested that computer game programming is essential and beneficial for engaging middle school children in training them for future computer classes and careers. Multiple attempts and innovations have been seen over the years. Several computing languages and programming applications have been developed to get students into programming since Logo was created in 1967 (e.g., Kelleher & Pausch, 2005; Kahn, 2007; Mc Nerney, 2004).

Game construction embodies the constructionist perspective that views learning as an active process in which the student actively forms knowledge by making things

(Rovegno & Dolly, 2006). When students develop games, they make interactive things and, in doing so, promote the building of knowledge. Constructionists' views aim to give students ways to develop their games and develop new links with knowledge instead of integrating lessons directly into games (Kafai, 2006). Papavlasopoulou et al. (2018) used the constructionist approach to developing a digital game using Scratch™ and found that students used specific programming concepts such as sequence/event handling and conditionals followed by threads and operators in developing the digital game. Scratch™ is a programming environment that visually represents programming concepts, and was developed to guide learners aged 8 years. Though aimed at students, the tool is frequently used to teach young and mature students basic programming principles (e.g., Malan & Leitner, 2007; Fadjo, Lu, & Black, 2009; Fadjo, Hallman, et al., 2009; Maloney et al., 2010) because programming concepts are visualised in blocks that students can snap together to create scripts (e.g., Resnick et al., 2003; Maloney et al., 2008).

Wilson et al. (2011) used Scratch™ for game making with students aged eight and nine years old to learn programming concepts, whilst others focused on 13–14-year-old (e.g., Adams, 2010; Papavlasopoulou et al., 2018; Sivilotti & Laugel, 2008) and university students (Malan, 2010). Grover et al. (2014) compared the learning performance of Scratch™ programming between Israeli middle school students and Northern California students; the findings indicate a significant difference in students' performance between pre-test and post-test. Scratch™ is also able to motivate young people to learn how to program (e.g., Brennan & Resnick, 2013; Maloney et al., 2010; Meerbaum-Salant et al., 2013; Resnick et al., 2009; Wolz et al., 2008) and allows beginners or novices to quickly create the games, animations, and interactive stories without the need to have basic programming syntax knowledge (Zaharija et al., 2013). Based on the findings and recommendations from the studies mentioned earlier, we decided that Scratch™ is the best tool to use.

### *Computational Thinking*

Learning programming concepts is almost always associated with general misconceptions about the tasks and the field (Clancy, 2004; Qian & Lehman, 2017; Kaczmarczyk et al., 2010). Problem-solving and CT are the keys to a programming language (Wong & Cheung, 2018), which requires concentrating on the syntax and semantics in understanding abstract concepts and learners must grasp its patterns of evidence (Kazimoglu et al., 2012). Educators have utilised various approaches to integrate CT concepts into computer programming, such as using computer game creation to engage students in the CT (Werner et al., 2012) or visual programming tools to teach programming, which is attractive and motivational (Kazimoglu et al., 2012); impact/foster creativity among students (Bennett et al., 2011); allow children to do more program manipulation (Rose et al., 2017). Visual programming tools are also frequently considered ideal as they enable learners to generate multiple abstractions quickly without the need for extreme program coding. Ioannidou et al.'s (2011) study

indicate an evident transfer between game design and science simulation design; it suggests that the CT components used to build the games could develop science simulations.

The Computer Science Teacher Association (CTSA) divided CT into six dimensions: formulating problems in a way that machines can help to solve, processing data in a logical way, representing data abstractly, algorithmising the automated solutions, and problems in an efficient way, and transferring knowledge and skills in solving other issues (CSTA, 2011), whilst Brennan and Resnick (2012) classified the CT into three constructs: computational concepts (sequence, loops, events, parallelism, conditionals, operators, data), computational practice (experimenting and interpreting, testing, and debugging, reusing, and mixing abstraction, and modulation) and computational perspectives (expressing, connecting, questioning).

However, Wing (2006) explained that CT integrates all vital skills related to problem-solving and proposed primary constructs of CT ranging from Abstraction and Decomposition; Representation; Problem reformation and Problem reformulation; Recursion; Parallelism; Generalisation; Systematic testing; Prevention, Protection, and Recovery. Similarly, Kazimoglu et al. (2012) proposed five core CT skills for a computer science course: problem-solving, algorithm building, debugging, simulation, and socialising whilst Selby and Woollard (2013) described the CT components as abstraction, evaluation, decomposition, algorithmic thinking, and generalisation. However, Romero et al. (2017) identified six CT competency components in the #5c21 framework model: problem identification, modelling, programming, and evaluation (related to Collaborative Problem Solving), code literacy and technological literacy.

Romero et al. (2016, 2017) identified five levels of learning to code activities as follows: (1) Instructor-centred descriptions and lessons; (2) Practical, step-by-step programming; (3) content creation programming individually; (4) co-creation content programming; (5) participatory co-creation of knowledge through programming. Students undergo a passive learning phase in typical learning, moving toward a more social constructivist experience. Drawing from these levels, we used the levels as markers in the experiment to determine how passive/active the participants were engaged in the assigned tasks. These levels inform the criteria for participant selection and how the actual investigation sessions are planned. Participants are led through each level of learning to code on the assumption that they have zero knowledge of programming and no available resources at home or school to learn to code. Participants are expected to work in teams to co-create actively at the end of the experience to illustrate their ability to collaborate on a programming task when guidance is reduced.

It is essential to evaluate the success of CT skills integration using computer language; thus, it is not simple to design the tools or measurements to assess CT. There have been various attempts to create CT measurement instruments. For instance, a framework to examine children's CT development through their ScratchTM programming products (Brennan & Resnick, 2012), a web-based tool to measure CT and programming skills called Dr ScratchTM (Moreno-León & Robles, 2015a, 2015b), and

a CT assessment framework (Seiter & Foreman, 2013). While numerous frameworks have been used to characterise the fundamental components of CT, these varied approaches to evaluating CT skills point to a propensity to concentrate on analysing the results of creative learning activities. The proposed definition of the CT words required the inclusion of concepts thoroughly defined in the literature.

## **Methodology**

### *Research Design*

Many CT research focuses on quantitative and experimental design in elementary, middle, high school or higher education institutions. However, this study uses the qualitative research method, which adopted the critical inquiry or transformative paradigm suggested by Riyami (2015) to understand the CT skills in co-creation activities amongst young sub-urban students in Malaysia. The researchers need to understand contemporary issues such as CT skills among underprivileged students from limited technological backgrounds (both at home and school) with zero knowledge/limited knowledge of programming language. In addition, they are less likely to be exposed to technical knowledge and skills such as digital games and animation, much less to the construct of such technology. This approach will help researchers understand how the co-creation activities of the unbalanced socio-economic students performed in such a disadvantaged society.

### *Participants*

A total of 36 students aged between 10–12 years old took part in this study (11 boys and 25 girls), and were recruited using a purposive sampling method. An email and social media were used to distribute a recruitment poster to parents living in a sub-urban district near the University. Participants were recruited based on several criteria, including residence and zero/limited programming knowledge. None of the participants was known to researchers, personally or professionally. The University's research ethics policies have been adequately followed throughout the research procedure because this study involved human subjects. Additionally, parental approval (consent) was requested.

The study was conducted in four separate sessions (one session of a pilot study and three sessions of the main study) within 4 months. The pilot study was conducted with five participants, while the remaining three sessions were conducted with 8–16 participants. They were further divided into smaller teams of two to four participants. Three researchers participated as facilitators during the experiment, while the other two participated as participant observers. It is essential to highlight that all facilitators were limited to guiding the construction process instead of giving instruction, feedback, and interventions during all sessions. Two of the facilitators are University lecturers with a background in animation and virtual reality, and another facilitator is a graduate

research student. The facilitator guided the participant to brainstorm the idea and provided assistance if problems arose during the co-creation process or if they had issues with the tool. Furthermore, the facilitators also acted if there were issues between the participants.

### *Apparatus and Materials*

IBM computer Laptop equipped with Scratch™ programming software and iSpring screen recorder software. Scratch™ was selected for this study because it represents a simplified visual approach to programming by combining the coloured command blocks to execute the 2-D graphical objects on the background screen called a stage. Also, the visual cues within Scratch™ were considered sufficient to provide ample input for the participants. Besides, it helps ease the novices in programming as the coloured code blocks have been categorised according to their function, such as variables, sounds, controls, and sensing. The Scratch game design tutorials were shown on the portable projector screen placed in front of the participants. The introductory tutorial modules used in this workshop include Getting Started Module, Chatbot Module, Memory Module, and Brain Game Module. Three video cameras and two GoPro were used to record the activities. These cameras were placed to surround the participants and were unobtrusive. In addition, each group were provided with a pen and paper.

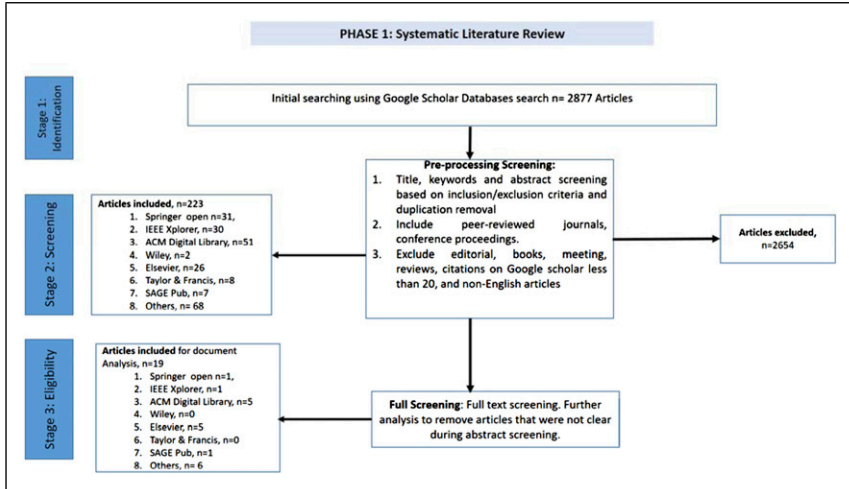
### *Development of Instrument*

The CT skills instruments in this study were developed by performing document analysis and can be divided into two main phases; (1) systematic literature review following guidelines by [Kitchenham \(2004\)](#) and reported and adapted using a PRISMA flow diagram as suggested by [Moher et al. \(2009\)](#) (refer to [Figure 1](#)), and (2) document analysis involving three researchers.

### *Phase 1: Systematic literature review analysis*

Phase 1 involved three main activities: Identification, Screening and Eligibility (included), as depicted in [Figure 1](#). During the process, the review protocol, the inclusion and exclusion criteria, and analysing of the relevant literature/database were determined. In this phase, an extensive literature review was conducted to analyse the current state of the art in CT, focusing on peer-reviewed articles using keywords such as Computational Thinking and Scratch Programming using Google Scholar and yielded 2877 articles. Furthermore, the literature review led to several databases/publishers where the topic is mainly published, such as ACM, IEEE Xplore Digital Library, ERIC, Wiley, ScienceDirect, and others (published between 2006 and 2021). Since our study is based on [Wing \(2006\)](#), we have decided that the search should begin with literature from 2006.





**Figure 1.** PRISMA flow diagram.

However, due to overwhelming results, we filtered the articles based on several criteria: peer-reviewed articles/conference proceedings; their availability (inaccessible articles were excluded), duplication (same authors published similar topic/content or the same article but was published with different journal issues, i.e. some articles were initially published online after acceptance in 2019 but was recently given a new volume and issue number in 2022), only articles with more than 20 citations in google scholar were included. Furthermore, we excluded editorial, books, meetings and reviews. Also, only articles in English were considered. We then used snowball sampling for the articles based on the key (critical) articles and further expanded our search. Furthermore, a quick skim of the articles was made to ensure they fit our focus.

As a result, the following key literature was identified (i.e. Anderson, 2016; Angeli et al., 2016; Atmatzidou & Demetriadis, 2016; Barr & Stephenson, 2011; Bers et al., 2014; Berland & Lee, 2011; Chen et al., 2017; Curzon et al., 2014; Dierbach et al., 2011; Grover, 2011; Lee et al., 2011; Kazimoglu et al., 2012; Perković et al., 2010; Repenning et al., 2016; Romero et al., 2017; Rose et al., 2017; Selby & Wollard, 2013; Shute et al., 2017; Wing; 2006). These also aid in the development of a study checklist of CT skills.

## Phase 2: Document Analysis

Phase 2 involved three researchers analysing the relevant literature eligible for analysis in phase 1 to produce a new checklist to ensure no overlapping CT skills on the final checklist.

Inter-rater reliability analysis was conducted during the process. Three researchers discussed and debated, and any conflicts were resolved during the group discussions until the final agreement was made. Based on key (critical) literature, we summarised the key findings from the analysis as illustrated in [Table 1](#).

Based on the CT Skills Model/framework/definition/concept documented in this phase ([Table 1](#)), we looked at the pattern in previous literature. Those CT Skills model/framework/definition/concepts with similar terms and definitions were merged and considered a category. As a result, 18 categories of CT skills emerged. We were combining the terms with similar definitions, which led to eliminating some CT Skills Model/framework/definition/concept, eventually revealing the 18 new CT skills as illustrated in [Appendix 1](#).

Subsequently, we have further analysed these CT skills ([Appendix 1](#)) to develop narrower CT skills to enable the mapping process between the skills and the digital game co-created by the children during the study. The justification for including or excluding 18 CT skills is based on definition, usage and similarity across the literature. The resulting terminology reflects the skills in the literature while eliminating those less clearly defined. The skills which had similar definitions were also unified. After several iterations of eliminating and finalising the CT skills, we have agreed that the final checklist of CT skills consists of ten core skills: abstraction, decomposition, algorithmic thinking, generalisation, representation, socialisation, code literacy, automation, co-ordination, and debugging, as illustrated in [Table 2](#).

For this study, we defined the ten core skills of CT as follows:

1. Abstraction: Break a problem into smaller parts to reduce complexity by removing unnecessary details
2. Decomposition: Break problems down by functionality
3. Algorithmic Thinking: Step-by-step procedures or instructions (commands)
4. Generalisations: A capability to make deductions from a specific to broader applicability.
5. Representation: Expressing problems and their workable solutions via a model or a formula.
6. Socialisation: Involve multiple parties with different resources during the process.
7. Code Literacy: Ability to perform computation due to programming language knowledge.
8. Automation: Ability to execute a set of repetitive tasks.
9. Coordination: Ability to control computational timing.
10. Debugging: Determining problems to fix malfunctioning rules and algorithms.

## **Data Analysis**

Our study has analysed each Scratch code created by participants to ensure that it can be mapped to the ten core CT skills checklist produced in the first stage of our studies.

**Table I.** Summarise of CT skills model/framework/definition/concept across literature.

| Literature                      | Number of CT Skills | Details of CT Skills, Model/Framework/Definition/Concept   |
|---------------------------------|---------------------|--|
| Dierbach et al. (2011)          | 4                   | Problem identification, algorithm building, model development, and evaluation  |
| Perković et al. (2010)          | 7                   | Computation (algorithm execution), communication (information transmission), coordination (computational timing control), Recollection (data organisation), automation, evaluation, design (abstraction, decomposition, system organisation) |
| Berland and Lee (2011)          | 5                   | Conditional logic, algorithm building, debugging, simulation, and distributed computation  |
| Lee et al. (2011)               | 3                   | Analysis, abstraction, and automation  |
| Kazimoglu et al. (2012)         | 5                   | Problem-solving, algorithm building, debugging, simulation, and socialising  |
| Selby and Wollard (2013)        | 5                   | Abstraction, decomposition, algorithmic thinking, evaluation, and generalisation   |
| Repenning et al. (2016)         | 3                   | Abstraction, analysis, and automation  |
| Angeli et al. (2016)            | 4                   | Abstraction, algorithm, decomposition, and generalisation  |
| Curzon et al. (2014)            | 5                   | Algorithmic thinking, decomposition, generalisation, abstraction, and evaluation   |
| Romero et al. (2017)            | 6                   | Problem identification, organisation/modelling, code literacy, technological literacy, programming, and evaluation   |
| Barr and Stephenson (2011)      | 9                   | Data collection, analysis, representation, problem decomposition, abstraction, algorithm & procedures, automation, Parallelization, and simulation   |
| Shute et al. (2017)             | 6                   | Debugging, iteration, algorithm, abstraction, decomposition, generalisation  |
| Bers et al. (2014)              | 3                   | Abstraction, generalisation, and, Trial and error activities   |
| Anderson (2016)                 | 5                   | Decomposition, pattern recognition, abstraction, algorithm design, evaluation  |
| Wing (2006)                     | 8                   | Abstraction and decomposition; representation; problem reformation and problem reformulation; Recursion; parallelism; generalisation; systematic testing; Prevention, Protection, and Recovery   |
| Chen et al. (2017)              | 5                   | Syntax, algorithm, data organisation, representation, Effectiveness/Efficiency   |
| Atmatzidou & Demetriadis (2016) | 5                   | Abstraction, generalisation, Modularity, decomposition, algorithm  |

*(continued)*

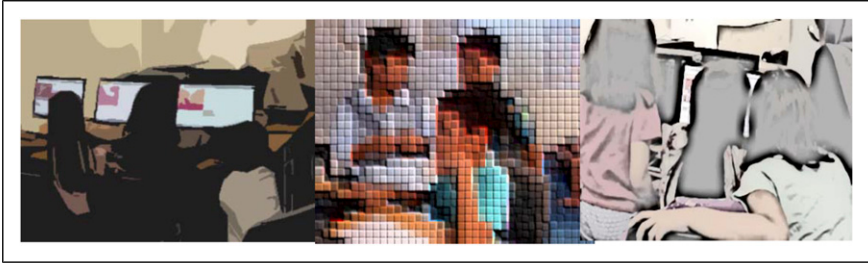
**Table 1.** (continued)

| Literature         | Number of CT Skills | Details of CT Skills, Model/Framework/Definition/Concept  |
|--------------------|---------------------|---|
| Grover (2011)      | 7                   | Computational thinking language (CTL), abstraction, task breakdown, conditional logic, representation, algorithm, and debugging   |
| Rose et al. (2017) | 7                   | Abstraction and generalisation; algorithm and procedures; data collection, analysis and representation; decomposition; parallelism; debugging, testing and analysis; control structures |

**Table 2.** Newly proposed CT skills and their justification.

| Newly Proposed CT    | Status (From Phase 2 Document Analysis, Table 1)  | Justification  |
|----------------------|---|--|
| Abstraction          | Keep the term (1)   | Keep the widely used term  |
| Decomposition        | Keep the term (2)   | Keep the widely used term  |
| Algorithmic thinking | Combined algorithmic thinking (4), conditional logic (9) and evaluation (18)                      | Keep the widely used term in literature and combine the conditional logic term as part of algorithmic thinking                             |
| Generalisation       | Keep the term (5)   | Keep the widely used term  |
| Representation       | Combined problem identification (3), Organising/ Modelling (6), representation (7), analysis (17) | Keep the most appropriate term to explain these CT skills  |
| Socialisation        | Rename the term distributed computation (10)  | Socialisation is a preferable term to explain the different terms (communication, socialising, distributed computation) used in literature |
| Code literacy        | Combined code literacy (11), technological system literacy (12), and programming (13)             | Keep the most appropriate term to explain these CT skills  |
| Automation           | Keep the term (14)  | Keep the relevant term   |
| Coordination         | Keep the term (15)  | Keep the relevant term   |
| Debugging            | Combined simulation (8) and debugging/iteration (16)  | Keep the most appropriate term to explain these CT skills  |

When narrative and observation data were collated during the co-creation, the focus was on identifying the CT skills demonstrated in actions, behaviours, and artefacts (a digital game). The co-creation process was observed using a video camera and a screen recorder, and all data was consolidated and analysed by five researchers (facilitators, participants' observers, and authors). Similar to the method in producing the checklist,



**Figure 2.** Co-creation of digital games during the pilot study.

the analysis was conducted in two stages, first individually and followed by a group. Any conflicts during the group discussions were resolved before the final mapping was made.

### *Pilot Study*

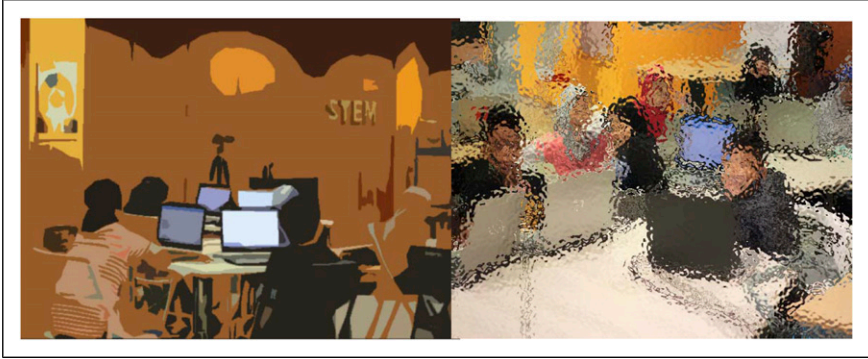
A pilot study was conducted with five students, and participants were divided into groups of three and two. The pilot study was conducted over 2 days (8 hours each day). The young children came from urban and suburban backgrounds. Three researchers acted as facilitators, while another two participated as participant observers. The pilot study was conducted to ensure the applicability of selected research apparatus and materials, such as the screen recorder software, camera, screen projector, and laptop arrangement. These coordination aspects are essential to ensure the data collected is usable and can be analysed. Also, the 10 CT skills checklist was used primarily in the pilot study to assess the instrument's validity and reliability.

Narrative data from the pilot study suggested that all 10 CT skills were visible during the co-creation activities (digital game artefacts and co-creation activities) as they were actively engaged (Figure 2). We had learned that some participants recruited for the pilot study already had a basic knowledge of Scratch™ before the session, which we did not anticipate. However, this has not affected the outcome as the pilot study aims to assess the instruments' validity and reliability. Furthermore, having some participants with basic knowledge of Scratch™ benefited the researchers as they provided excellent input for consideration in the next stage of the experiment. Hence we strictly enforced the requirements of no basic Scratch™ knowledge in the main study. These findings lead to the improvements such as the reliability of screen recorder software in recording screen activities and other physical arrangements, such as the room's layout for experimentation. Participants' feedback was also considered mainly on the length of the sessions, as the allocated time was too lengthy. Hence, the actual experiment was shortened to only 1 day per session instead of two consecutive days.

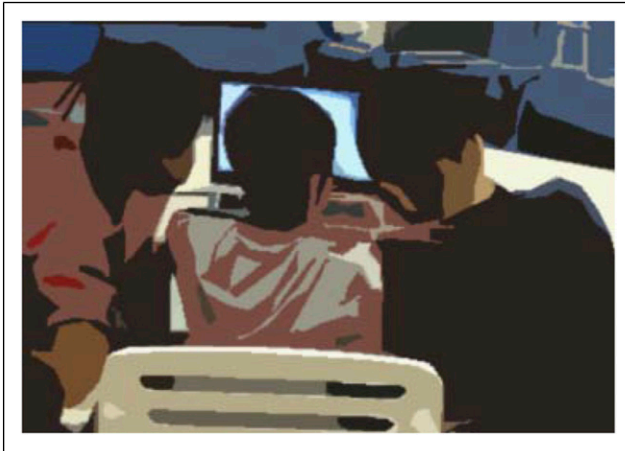
## **Research Procedure**

To ensure sufficient data about CT skills patterns while the young participants co-construct their digital games, the procedures to conduct each session were curated to be replicated.

1. **Briefing Session:** When the participants arrived with their parents, they were briefed about the project. Participants were divided into groups of two to four, and parents and participants were asked to complete the consent form. Participants were asked to sit at the designated table randomly; hence, there is a mixture of group formation, all-female, all-male, and mixed genders.
2. **Ice-Breaking Session:** Participants were introduced to computer programming concepts and asked questions to trigger interest in learning programming concepts. For example, who likes to play games? What kind of games do you play? Who wants to create their games? It aims to warm them toward constructing digital games. The session was conducted in a small group activity to facilitate the collaborative process and help them get to know them. One instructor was assigned to each group to facilitate the session like an ice-breaking session.
3. **Design Workshop on Scratch™ (Training Session):** The participants were introduced to Scratch™ programming for 3 hours. Each participant was provided with an IBM laptop equipped with Scratch™ 2.0 software. Although only 3 hours of Scratch™ programming training was given to the participants, it covered all the ten CT skills needed in this study. They were introduced to the different features of the Scratch software to ensure they could demonstrate the ten CT skills required and co-create a digital game during the competition session. Also, the laptop was equipped with screen recording software to record participants' activities throughout the session. The iSpring Free Cam screen recorder software was installed on the laptop to observe the co-designing of the digital game among the participants and the digital game coding designed by the participants. Similar to the ice-breaking session, one instructor was assigned to each group to facilitate the session. Each participant was equipped with a laptop during this session, as shown in [Figure 3](#).
4. **Design Workshop on Digital Games Creation Session (Competition Session):** They were assigned to work as a team to challenge the participants. A timeline was given to ensure they could achieve measurable steps throughout the co-creation process. Each team produced a digital game and was provided with necessary tools, such as stationery for the brainstorming session and storyboard creation [Figure 4](#). Also, the facilitators guided the participants in co-creating the digital game. Only one laptop per group was given during this session to ensure the interaction between participants occurred on a single screen, as illustrated in [Figure 4](#). The idea and storyboard created at this stage were then translated into a digital



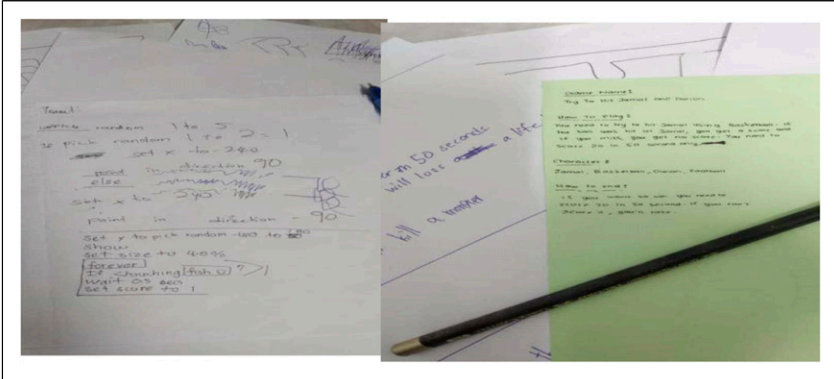
**Figure 3.** Participants' arrangements during the training session.



**Figure 4.** Participants' arrangements during brainstorming sessions and storyboard creation.

game using Scratch™. Examples of ideas and sketches made by some participants during the co-creation activities are shown in [Figure 5](#).

5. Debriefing Session: We used specific questions to ask participants about their experience constructing the games. Participants are informed of their contribution to understanding how young children co-create digital games and we thank them for their participation.



**Figure 5.** The participants' ideas and sketches using the stationery.

## Findings

### Descriptive Analysis

A total of 36 children participated in this study, including 11 boys and 25 girls. The children's ages ranged from 10 to 12 years old, with a mean age of 11.14 years (SD = 0.93).

### CT Skills Analysis

We analysed the game artefacts and co-creation process to answer Objective 2. The digital games created were analysed using the 10 CT skills checklist and can be summarised in [Table 3](#). The composition of each team, digital game completion status, and digital game types are also illustrated in [Table 3](#). [Table 3](#) summarises the different CT skills achieved by participants in this study, and three CT skills of socialisation, debugging, and algorithmic thinking were visible in each group.

Overall, four groups of participants demonstrated eight and seven CT skills; two groups demonstrated six skills, respectively, while two demonstrated five CT skills. None of the groups demonstrated between nine and ten CT skills. Nine groups completed their game within the time allocated, whilst three did not finish their game. [Table 3](#) also shows that all students in the male group demonstrated 7 CT skills, whilst all the female group demonstrated between 6 and 8 CT skills. However, the mixed-gender group demonstrated between 5 and 8 CT skills.

Additionally, four groups developed dual-player digital games, and eight developed single-player digital games. The digital games created by the participants were analysed by looking at the Scratch™ programming codes and were mapped with the nine CT skills (refer to [Table 4](#)), whilst one CT skill (socialisation) was analysed through video



**Table 3.** Summary of CT skills used in digital games' co-creation.

| The Session,<br>Group, and<br>Number of<br>Participants | CT Skills Demonstrated |    |   |   |   |   |   |    |    |   | Game<br>Completion<br>(Yes/No) | Digital Games<br>Type (Single or<br>Dual Player) |               |
|---|------------------------|----|---|---|---|---|---|----|----|---|--------------------------------|--|---------------|
|   | AB                     | DC | G | R | S | C | D | CL | AT | A |                                |  |               |
| <b>SESSION 1</b>  |                        |    |   |   |   |   |   |    |    |   |                                |  |               |
| SI, G1 (3M)   | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Dual player   |
| SI, G2 (3F)   | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| SI, G3 (2F, 1M)   |                        |    | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| SI, G4 (3F)   |                        | /  | / | / | / | / | / | /  | /  | / | /                              | No   | Single-player |
| <b>SESSION 2</b>  |                        |    |   |   |   |   |   |    |    |   |                                |  |               |
| S2&G1 (3F)  |                        | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| S2&G2 (3M)  | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| S2&G3 (4F)  |                        |    | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| S2&G4 (2F, 1M)  |                        | /  | / | / | / | / | / | /  | /  | / | /                              | No   | Dual player   |
| S2&G5 (3F)  | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Single-player |
| <b>SESSION 3</b>  |                        |    |   |   |   |   |   |    |    |   |                                |  |               |
| S3&G1 (1M, 2F)  | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Dual player   |
| S3&G2 (3F)  | /                      | /  | / | / | / | / | / | /  | /  | / | /                              | No   | Single-player |
| S3&G3 (2M)  |                        | /  | / | / | / | / | / | /  | /  | / | /                              | Yes  | Dual player   |

recording of co-creation activities. Interestingly, the two groups who demonstrated the most CT skills did not complete their digital game.

Although we allowed participants to sit and form the group randomly, one issue with collaboration appeared in one group throughout the process. During the discussion, one of the children in the red shirt pushed her chair away, facing a different direction from the other group members, as shown in [Figure 6](#) (left). One of the facilitators quickly intervened, and they completed their digital game despite collaboration issues. Surprisingly, the group attained 8 CT skills.

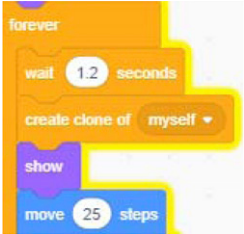

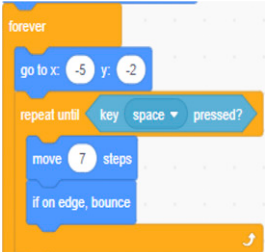

[Table 5](#) shows an example of digital games created by two teams. These examples illustrate the different levels of CT skills attained during their co-creation activities.

## Discussion

### CT Skills Checklist

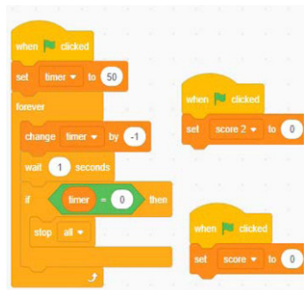
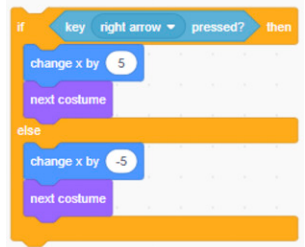
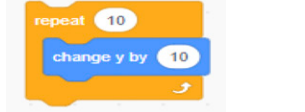
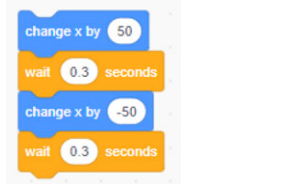

This study has successfully achieved its first aim to develop a new instrument (CT skills checklist) for learning programming concepts using document analysis. Our analysis of the recent literature on CT skills has provided an understanding of various efforts to describe and categorise CT skills. However, various researchers' use of different terms or definitions made the classification difficult. For example, the widely used term in CT is abstraction and is defined by [Wing \(2008\)](#) as deciding what details we need to

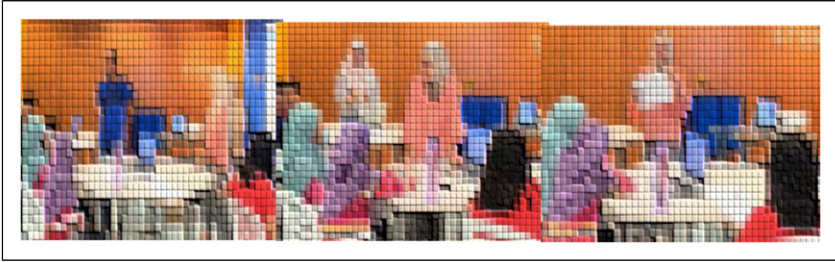
**Table 4.** Mapping of CT skills with digital games coding.

| CT Skills            | Digital Game Co-Design Task   | Digital Games Coding Extract   |
|----------------------|---|--|
| Abstraction          | Creating clones performing similar actions and behaviour  |     |
| Decomposition        | Decompose the parts of the game design into several parts, which can be the character's life-like motion and rules about how users can interact with the characters |    |
| Algorithmic thinking | Writing instructions to execute a set of blocks several times depends on the given situation to achieve the desired effect Using the repeat until block             |     |
| Generalisation       | The student's understanding of the axis allows them to apply it to the characters' movements  |  |

*(continued)*

**Table 4.** (continued)

| CT Skills      | Digital Game Co-Design Task   | Digital Games Coding Extract   |
|----------------|---|--|
| Representation | Storing information on a variable to modify the attributes of the characters in solving a problem   |    |
| Code literacy  | Apply interactivity by moving the characters, which can help to shape and reshape information about the characters. The use of conditional logic, the if-then-else construct  |    |
| Automation     | The repeat block instructs the computers to execute the repetitive task   |    |
| Coordination   | Synchronise characters to make them perform specific behaviour in the intended order. The use of the wait until block in designing the game   |   |
| Debugging      | Use the when green flag clicked block to recognise any possible errors in the logic formed. The debugging process checks if the rules or algorithms work or malfunction and compare them with the digital games created |  |



**Figure 6.** Collaboration issues (left) and how the instructor fosters collaboration within the group (middle and right).

**Table 5.** Different examples of digital games created and their description.

Highest CT Skills Demonstrated (8 Skills)

Screenshot



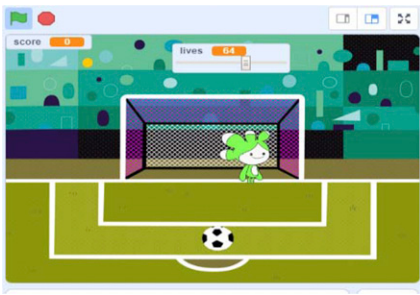
Game description

*(Dodge, the Goalie)*

This game was not successfully constructed since the original notion enabled dual players, but a player must control both balls. The player must shoot both balls without touching the goalkeeper. The player had to be aware of their movements and find the best time to shoot the ball because both the ball and the goalkeeper were constantly moving

Lowest CT skills demonstrated (5 skills)

Screenshot



Game description

*(Goal)*

The student did not complete the game. However, the idea was to let the player shoot the ball into the goal without touching the goalkeeper (a constantly moving Avatar). Thus, the score gained will depend on the success in shooting the ball into the goal. The default lives for each player are 64

highlight and what details we can ignore. However, [Perković et al. \(2010\)](#) explained in their framework that the design is an organisation (using abstraction, modularisation, aggregation, decomposition) of a system, process, object, etc. whilst it was viewed as a part of the design principle' simplicity by [Dennings \(2003\)](#). [Angeli et al. \(2016\)](#) described abstraction as the skill to decide what information about an entity/object to keep and what to ignore based on [Wing' \(2006\)](#).

We further consolidated these different terms and definitions and provided a new checklist to understand CT skills better. It is argued that the final checklist is valid as three researchers discussed and debated, and any conflicts were resolved during the group discussions until the final agreement was made. Furthermore, our findings show that this checklist is adequate and valuable in mapping CT skills for co-creation activities, although none of the groups exhibited all ten skills.

### *Instrument Validation Through Co-Creation Activities*

A recent literature review by [Tikva and Tambouris \(2020\)](#) provided a comprehensive analysis of different methods used to assess CT and disagreed on what and how to evaluate CT. They further explained that pre-test/post-test, observation, self-report, and artefact analysis are widely used. Furthermore, [Fagerlund et al. \(2021\)](#) summarised the various studies that involved different methods, contents/activities, and taxonomy/rubrics for assessing Scratch programming in K-9. However, our study combined artefact (digital game) analysis and observation in analysing the CT during co-creation activities. The digital games were analysed to determine the CT skills utilised by participants as they co-created digital games. We have selected two digital games ([Table 5](#)) which portray the most and least used CT skills to explain the different CT skills utilised in the co-creation activities.

Scratch™ programming software allows participants in this study to actively co-create digital games. It provides an enjoyable and straightforward way of creating digital games because it does not require learners to grasp all the basic ideas of programming constructs. However, although none of the groups demonstrated all ten CT skills in the co-creation of digital games, all ten CT skills were present in their digital games. This can be related to [Sung et al. \(2010\)](#), which explained that game creation/development requires significant knowledge of computer graphics and digital game-play; and time and effort to develop the digital game. Furthermore, this study also suggested that the performance of students between gender are varied and cannot be generalised based on the CT skills attained. These findings are parallel with [Ardito et al. \(2020\)](#), [Anuar et al. \(2020\)](#) and [Atmatzidou and Demetriadis \(2016\)](#), which suggested that the boys, girls or mixed-gender groups have their strengths and weaknesses in CT skills and further analysis are required to understand this phenomenon.

It is also vital to highlight in this study that, unlike other CT skills, socialisation skills were observed during the co-creation process, and all the groups could show these skills. In addition, participants could discuss and integrate different ideas during the co-creation process. In this study, participants were tasked with designing a functional and

interactive digital game using these programming constructs. [Perković et al. \(2010\)](#) described the process as crucial, which defines the computational process involving implementing algorithms via a sequence of phases until the goal is achieved. It is connected to one of the CT skills discussed in the result section, algorithmic thinking, where the participants must provide step-by-step processes to achieve a goal. The results showed that all groups were able to demonstrate algorithmic thinking skills. When the participants find ways to achieve the goal, it also promotes their ability to think in abstraction and decomposition. Participants, for example, can divide problems into smaller parts to reduce their complexity; thus, they can consider an abstraction.

In this study, participants practised abstractions when creating clones performing similar behaviour and actions during co-creation. They executed the intended instructions by clicking on a green flag icon at the top of the Scratch™ design interface. This feature of the Scratch™ programming tool enables participants to detect the issues to be fixed. This action of validating the abstractions has resulted in debugging, one of the CT skills in this study. Participants could assess the potential faults in their solutions using the debug key to obtain the most efficient and effective combination of steps and resources ([Kazimoglu et al., 2012](#)). During co-creation in this study, participants practised abstractions by making clones that exhibited comparable behaviour and actions. By tapping the green flag icon at the top of the Scratch™ design interface, they carried out the expected instructions. Participants can identify the problems that need to be fixed thanks to this function of the Scratch™ programming tool. Debugging, one of the CT abilities in this study is the outcome of the abstractions being validated. Using the debug key, participants could evaluate potential flaws in their ideas and develop the best possible workflow and resource allocation ([Kazimoglu et al., 2012](#)). The given debugging mechanism aids in the participants' analysis of the issue. It encourages them to use their abstraction skills, as Scratch™ offers automatic feedback through direct visual simulation of their solutions rather than technical programming terms.

Furthermore, incorporating CT into teaching and learning to promote problem-solving skills must be considered a mental process. The potential continuation of this study would consider the allocation of intervention time and see how it will impact students' performance and promote CT skills. Although the CT skills attained amongst the participants are diverse, it demonstrates that co-creation of the digital game can positively promote students' CT skills using Scratch™ programming software. Thus, it provides a perspective on CT skills regarding whether they should be restricted to programming or computer science. A recent study by [Chang et al. \(2018\)](#) suggested that education should be re-conceptualised and take an alternative view of CT, as computation is essential in all areas, not limited to computer science or programming. It is evident from this study that although none of the groups demonstrated all ten CT skills, they could develop digital games successfully in a short time and how these CT skills can be analysed beyond programming constructs.

## *Conclusion and Future Work*

Various instruments are available to measure CT skills, and this study extensively reviewed and consolidated existing instruments; and subsequently proposed ten CT skills: abstraction, decomposition, algorithmic thinking, generalisation, representation, socialisation, code literacy, automation, coordination, and debugging. This checklist can help future researchers analyse CT skills for learning programming concepts without using multiple instruments or checklists. Subsequently, we validate the checklist with the digital game's artefacts developed by suburban children. This study provides empirical proof of how young children create digital games using their CT skills. The co-creation of digital games with students has gotten less attention, despite the fact that utilising digital games to measure CT skills is widespread in the literature and it has been hypothesised that it aids in the development of CT. Therefore, it is crucial to ascertain whether CT development utilising Scratch™ programming tools is significantly impacted by the co-creation of digital games.

The work presented in this study validated the proposed ten CT skills identified through document analysis. It can be considered the most plausible way to measure the digital games co-created with students using Scratch™. This study provides important insights into CT focusing on sub-urban children with no programming knowledge.

Furthermore, although the collaboration issues between group members are only visible in one group, future work must foster collaboration between team members before moving to co-creation activities to overcome the issues highlighted in this study. However, in this study, we can mitigate the problem as it happened during the brainstorming session.

This study's generalizability may be hampered by one of its limitations, which was the small number of participants. In order to create a comprehensive description of CT capabilities, additional generalizable studies including more participants are required in the future. However, because the instrument (CT skills checklist) employs the qualitative method, the current sample size is adequate to validate it.

Future research will concentrate on validating the instruments with bigger groups and groups from different socioeconomic backgrounds to assess their abilities and potential to employ CT skills and to better understand how CT skills manifest themselves or evolve through co-creation activities. Finally, a research extension would develop a pedagogical framework enabling educators to integrate computer skills into subjects other than computer science, so broadening and enhancing CT skills across the national curriculum.

## Appendix I

### New Categories of CT Skills.

| Newly Proposed CT | Definition  | CT Skills and Article  |
|-------------------|---|--|
| Abstraction       | Breaks a problem into smaller components that are easier to understand, program, and debug in solving problems. Also, a process of generalisation from specific instances | Abstraction (Selby & Woollard, 2013)<br>Abstraction (Lee et al., 2011)<br>Design (Perković et al., 2010)<br>Abstraction (Curzon et al., 2014)<br>Abstraction (Rose et al., 2017)<br>Abstraction (Wing, 2006)<br>Abstraction (Barr & Stephenson, 2011)<br>Abstraction (Shute et al., 2017)<br>Abstraction (Bers et al., 2014)<br>Abstraction (Anderson, 2016)<br>Abstraction (Atmatzidou & Demetriadis, 2016)<br>Abstraction (Shute et al., 2017)<br>Abstraction (Grover, 2011) |
| Decomposition     | A process of breaking problems down by functionality, especially for complex problems and tasks   | Decomposition (Selby & Woollard, 2013)<br>Decomposition (Curzon et al., 2014)<br>Problem decomposition (Barr & Stephenson, 2011)<br>Decomposition (Shute et al., 2017)<br>Decomposition (Anderson, 2016)<br>Decomposition (Wing, 2006)<br>Decomposition (Atmatzidou & Demetriadis, 2016)<br>Decomposition (Shute et al., 2017)<br>Task breakdown (Grover, 2011)<br>Decomposition (Rose et al., 2017)   |

(continued)



(continued)

| Newly Proposed CT      | Definition  | CT Skills and Article  |
|------------------------|---|--|
| Problem identification | The capability to recognise the parts of a situation and its structure. Also, the process of analysing and representing the situation encountered | Syntax (Chen et al., 2017)<br>Problem identification (Romero et al., 2017)<br>Problem identification (Dierbach et al., 2011)<br>Problem-solving (Kazimoglu et al., 2012)<br>Data collection (Barr & Stephenson, 2011)<br>Pattern Recognition (Anderson, 2016)<br>Problem reformulation and reformation (Wing, 2006)<br>Modularity (Atmatzidou & Demetriadis, 2016)<br>Data collection, analysis and representation (Rose et al., 2017)   |
| Algorithmic thinking   | A step-by-step procedure or instructions (commands) to accomplish a task  | Paralllism (Rose et al., 2017)<br>Algorithmic thinking (Selby & Woollard, 2013)<br>Algorithm building (Berland & Lee, 2011)<br>Computation (Perković et al., 2010)<br>Algorithm building (Dierbach et al., 2011)<br>Algorithm building (Kazimoglu et al., 2012)<br>Algorithmic thinking (Curzon et al., 2014)<br>Algorithm & procedures (Barr & Stephenson, 2011)<br>Algorithms (Shute et al., 2017)<br>Algorithm design (Anderson, 2016)<br>Algorithm (Atmatzidou & Demetriadis, 2016)<br>Parallelization (Barr & Stephenson, 2011)<br>Algorithm (Grover, 2011)<br>Algorithm and procedures (Rose et al., 2017)<br>Recursive (Wing, 2006)<br>Parallelism (Wing, 2006)<br>Algorithms (Chen et al., 2017) |

(continued)

(continued)

| Newly Proposed CT       | Definition  | CT Skills and Article  |
|-------------------------|---|--|
| Generalisation          | The capability to make deductions from a specific to broader applicability  | Generalisation (Shute et al., 2017)<br>Generalisation (Selby & Woollard, 2013)<br>Generalisation (Curzon et al., 2014)<br>Generalisation (Bers et al., 2014)<br>Generalisation (Atmatzidou & Demetriadis, 2016)<br>Generalisation (Rose et al., 2017)<br>Generalisation (Wing, 2006) |
| Organising/Modelling    | The capability to manage and represent the situation proficiently   | Recollection (Perković et al., 2010)<br>Organising/ Modelling (Romero et al., 2017)<br>Data Organization (Chen et al., 2017)<br>Models development (Dierbach et al., 2011)   |
| Representation          | A modelling process demonstrates problems and solutions using different ways, such as a replica or formula  | Representation (Chen et al., 2017)<br>Representation (Barr & Stephenson, 2011)<br>Representation (Grover, 2011)<br>Representation (Wing, 2006)   |
| Simulation              | A process of modelling or assessing algorithms or logic. It is applied in debugging to find problems and uses algorithm building to test a model. It is also defined as the representation of algorithms or plans | Simulation (Barr & Stephenson, 2011)<br>Simulation (Berland & Lee, 2011)<br>Simulation (Kazimoglu et al., 2012)  |
| Conditional logic       | Use of the if-then-else concept. It needs a student to reason at a macro level about the outcome of the truth-value of the statement  | Conditional logic (Berland & Lee, 2011)<br>Conditional logic (Grover, 2011)<br>Control structures (Rose et al., 2017)  |
| Distributed computation | An application of rule-based actions. The contingencies and strategy formation will include numerous groups with diverse knowledge resources  | Socialising (Kazimoglu et al., 2012)<br>Distributed computation (Berland & Lee, 2011)<br>Communication (Perković et al., 2010)   |
| Code literacy           | A capability to perform computation because of having programming language knowledge  | Code literacy (Romero et al., 2017)  |

(continued)

(continued)

| Newly Proposed CT              | Definition   | CT Skills and Article   |
|--------------------------------|--|---|
| Technological system literacy  | Having knowledge and practical familiarity with hardware, software, peripherals, and network components related to information systems | Technological literacy (Romero et al., 2017)<br>Computational thinking language (CTL) Grover (2011)   |
| Programming                    | Skills to make a computer program  | Programming (Romero et al., 2017)   |
| Automation                     | A process of performing a group of recurring tasks rapidly and effectively   | Automation (Lee et al., 2011)<br>Automation (Barr & Stephenson, 2011)<br>Automation (Perković et al., 2010)   |
| Coordination (synchronisation) | Ability to control computational timing  | Coordination (Perković et al., 2010)  |
| Debugging/ Iteration           | It is determining problems to fix malfunctioning rules and algorithms. It is an iterative process                                      | Debugging (Kazimoglu et al., 2012)<br>Iteration (Shute et al., 2017)<br>Trial and Error activities (Bers et al., 2014)<br>Debugging (Shute et al., 2017)<br>Debugging (Berland & Lee, 2011)<br>Debugging (Grover, 2011)<br>Prevention, protection and recovery (Wing, 2006)                               |
| Analysis                       | A reflective practice to validate whether the abstractions made are correct  | Analysis (Lee et al., 2011)<br>Analysis (Barr & Stephenson, 2011)   |
| Evaluation                     | Evaluating the effectiveness and the efficiency of algorithmic processes   | Evaluation (Selby & Woollard, 2013)<br>Evaluation (Romero et al., 2017)<br>Effectiveness/Efficiency (Chen et al., 2017)<br>Evaluation (Perković et al., 2010)<br>Evaluation (Dierbach et al., 2011)<br>Evaluation (Curzon et al., 2014)<br>Evaluation (Anderson, 2016)<br>Systematic testing (Wing, 2006) |

## Acknowledgments

We gratefully acknowledge the grant from Universiti Malaysia Sarawak (F04/SpMYRA/1657/2018).

## Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study is supported by Universiti Malaysia Sarawak (F04/SpMYRA/1657/2018).

## ORCID iDs

Mohd Kamal Othman  <https://orcid.org/0000-0001-5401-2515>

Fitri Suraya Mohamad  <https://orcid.org/0000-0003-4460-8061>

## References

- Adams, J. C. (2010). Scratching middle schoolers' creative itch. In *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM. (pp. 356-360). <https://doi.org/10.1145/1734263.1734385>
- Anderson, N. D. (2016). A call for computational thinking in undergraduate psychology. *Psychology Learning & Teaching, 15*(3), 226–234. <https://doi.org/10.1177/1475725716659252>
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for teacher knowledge. *Journal of Educational Technology & Society, 19*(3), 47–57.
- Anuar, N. H., Mohamad, F. S., & Minoi, J. L. (2020). Art-integration in computational thinking as an unplugged pedagogical approach at A rural Sarawak primary school. *International Journal of Academic Research and Social Sciences, 10*(17), 21–39. <http://dx.doi.org/10.6007/IJARSS/v10-i17/8328>
- Ardito, G., Czerkawski, B., & Scollins, L. (2020). Learning computational thinking together: Effects of gender differences in collaborative middle school robotics program. *TechTrends, 64*(3), 373–387. <https://doi.org/10.1007/s11528-019-00461-8>
- Armstrong, D., Gosling, A., Weinman, J., & Marteau, T. (1997). The place of inter-rater reliability in qualitative research: An empirical study. *Sociology, 31*(3), 597–606. <https://doi.org/10.1177/0038038597031003015>
- Arnab, S., Clarke, S., & Morini, L. (2019). Co-creativity through play and game design thinking. *Electronic Journal of e-Learning, 17*(3), 184–198. <https://doi.org/10.34190/JEL.17.3.002>
- Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670. <https://doi.org/10.1016/j.robot.2015.10.008>
- Balanskat, A., & Engelhardt, K. (2014). Computing our future: Computer programming and coding-priorities. *School Curricula and Initiatives Across Europe. European Schoolnet, 3*(50), 1.
- 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>
- Bennett, V. E., Koh, K. H., & Repenning, A. (2011). CS education Re-kindles creativity in public schools. In *Proceedings of the 16th annual joint conference on innovation and technology in*

- computer science education. ACM. (pp. 183–187). <https://doi.org/10.1145/1999747.1999800>
- Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning (IJGBL)*, 1(2), 65–81. <https://doi.org/10.4018/ijgbl.2011040105>
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72, 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>
- Blummer, B. (2008). Digital literacy practices among youth populations: A review of the literature. *Education Libraries: Children Resources*, 31(1), 38–45. <https://doi.org/10.26443/el.v3i1i3.261>
- Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 215–220). <https://doi.org/10.1145/2445196.2445265>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association* (p. 25).1
- Brennan, K., & Resnick, M. (2013). Stories from the scratch community: Connecting with ideas, interests, and people. *Proceedings of the 44th ACM technical symposium on Computer science education '13*. ACM (pp. 463–464).
- Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK. *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM. (pp. 269-274) <https://doi.org/10.1145/2445196.2445277>
- Chang, Z., Sun, Y., Wu, T.Y., & Guzaini, M. (2018). Scratch analysis tool (SAT): A modern scratch project analysis tool based on ANTLR to assess computational thinking skills. In *14th international wireless communications & mobile computing conference (IWCMC)* (pp. 950–955). IEEE. <https://doi.org/10.1109/IWCMC.2018.8450296>
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- Clancy, M. (2004). Misconceptions and attitudes that interfere with learning to program. In S. Fincher & M. Petre (Eds), *Computer science education research* (pp. 95–110).
- CSTA (2011). *Operational definition of computational thinking for Ke12 education*. CSTA. <http://www.csta.acm.org/Curriculum/sub/CompThinking.html>
- Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing computational thinking in the classroom: A framework, swindon, gb*. <http://eprints.soton.ac.uk/id/eprint/369594>
- de Araujo, A. L. S. O., Andrade, W. L., & Guerrero, D. D. S. (2016). A systematic mapping study on assessing computational thinking abilities. In *IEEE frontiers in education conference (FIE)* (pp. 1–9). IEEE. <https://doi.org/10.1109/FIE.2016.7757678>

- Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, 58(1), 240–249. <https://doi.org/10.1016/j.compedu.2011.08.006>
- Denning, P. J. (2003). Great principles of computing. *Communications of the ACM*, 46(11), 15–20. <https://doi.org/10.1145/948383.948400>
- Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., et al (2011). A model for piloting pathways for computational thinking in a general education curriculum. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM (pp. 257–262 <https://doi.org/10.1145/1953163.1953243>)
- Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research*, 60(2), 481–511. <https://doi.org/10.1177/07356331211033158>
- Fadjo, C. L., Hallman, G. Jr, Harris, R., & Black, J. B. (2009b). Surrogate embodiment, Mathematics instruction and video game programming. *EdMedia+ innovate learning* (pp. 2787–2792). Association for the Advancement of Computing in Education (AACE).
- Fadjo, C. L., Lu, M. T., & Black, J. B. (2009a). *Instructional embodiment and video game programming in an after school program EdMedia+ innovate learning*. Association for the Advancement of Computing in Education (AACE) (pp. 4041–4046).
- Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12–28. <https://doi.org/10.1002/cae.22255>
- García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, 80, 407–411. <https://doi.org/10.1016/j.chb.2017.12.005>
- García-Peñalvo, F. J., Rees, A. M., Hughes, J., Jormanainen, I., Toivonen, T., & Vermeersch, J. (2016). A survey of resources for introducing coding into schools. In *Proceedings of the fourth international conference on technological ecosystems for enhancing multiculturalism* (pp. 19–26). ACM. <https://doi.org/10.1145/3012430.3012491>
- Grover, S. (2011). *Robotics and engineering for middle and high school students to develop computational thinking*. Annual meeting of the American educational research association.
- Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 57–62). <https://doi.org/10.1145/2591708.2591713>
- 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>
- Guzdial, M. (2008). Education: Paving the way for computational thinking. *Communications of the ACM*, 51(8), 25–27. <https://doi.org/10.1145/1378704.1378713>
- Hambusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multi-disciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, 41(1), 183–187. <https://doi.org/10.1145/1539024.1508931>
- Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H., & Basawapatna, A. (2011). *Computational thinking patterns*. Annual meeting of the American educational research association. . <https://files.eric.ed.gov/fulltext/ED520742.pdf>

- Jiang, B., Zhao, W., Gu, X., & Yin, C. (2021). Understanding the relationship between computational thinking and computational participation: A case study from scratch online community. *Educational Technology Research and Development*, 69(5), 2399–2421. <https://doi.org/10.1007/s11423-021-10021-8>
- Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 107–111). ACM. <https://doi.org/10.1145/1734263.1734299>
- Kafai, Y. B. (2006). Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture*, 1(1), 36–40. <https://doi.org/10.1177/1555412005281767>
- Kahn, K. (2007). Should LOGO keep going forward 1? *Informatics in Education-An International Journal*, 6(2), 307–321. <https://doi.org/10.15388/infedu.2007.20>
- Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital gameplay. *Procedia Computer Science*. (9), 522–531. <https://doi.org/10.1016/j.procs.2012.04.056>
- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2), 83–137. <https://doi.org/10.1145/1089733.1089734>
- Kitchenham, B. (2004). *Procedures for performing systematic reviews*. : Keele University Technical Report TR/SE-0401Keele University.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Malyn-Smith, J., & Werner, L. (2011). Ericks on Computational Thinking for youth in practice. *ACM Inroads*, 2(1), 33–37. <https://doi.org/10.1145/1929887.1929902>
- Lye, S. Y., & Koh, J. H. L. (2014). Review of 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. (2010). Reinventing CS50. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 152–156). ACM. <https://doi.org/10.1145/1734263.1734316>
- 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>
- Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch Programming Language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 1–15. <https://doi.org/10.1145/1868358.1868363>
- Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with scratch. *SIGCSE'08*, 40(1), 367–371. <https://doi.org/10.1145/1352135.1352260>
- McDonald, N., Schoenebeck, S., & Forte, A. (2019). Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. In *Proceedings of the ACM on human-computer interaction* (3). ACM. <https://doi.org/10.1145/3359174>

- McNerney, T. S. (2004). From turtles to tangible programming bricks: Explorations in physical language design. *Personal and Ubiquitous Computing*, 8(5), 326–337. <https://doi.org/10.1007/s00779-004-0295-6>
- 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>
- Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *PLoS Medicine*, 6(7), e1000097. <https://doi.org/10.1371/journal.pmed.1000097>
- Moreno-León, J., & Robles, G. (2015a). *Analyse your scratch projects with Dr Scratch and assess your computational thinking skills*. Scratch conference (pp. 12–15).
- Moreno-León, J., & Robles, G. (2015b). Dr Scratch: A web tool to automatically evaluate scratch projects. In *Proceedings of the workshop in primary and secondary computing education* (pp. 132–133). <https://doi.org/10.1145/2818314.2818338>
- Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development*, 68(1), 463–484. <https://doi.org/10.1007/s11423-019-09708-w>
- Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2018). Discovering children’s competences in coding through the analysis of Scratch projects. In *IEEE global engineering education conference (EDUCON)* (pp. 1127–1133). IEEE. <https://doi.org/10.1109/EDUCON.2018.8363356>
- Perković, L., Settle, A., Hwang, S., & Jones, J. A. (2010). A framework for computational thinking across the curriculum. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 123–127).
- Qian, Y., & Lehman, J. (2017). Students’ misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1), 1–24. <https://doi.org/10.1145/3077618>
- Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, 25(5), 66–71.
- Repenning, A., Basawapatna, A., & Escherle, N. (2016). Computational thinking tools. In *IEEE symposium on visual languages and human-centric computing (VL/HCC)* (pp. 218–222). IEEE. <https://doi.org/10.1109/VLHCC.2016.7739688>
- Resnick, M., Kafai, Y., & Maeda, J. (2003). *A networked. Media-rich programming environment to enhance technological fluency at after-school centers in economically disadvantaged communities*. Proposal to the National Science Foundation.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al (2009). Scratch: Programming for all. *Communication ACM*, 52(11), 60–67. <https://doi.org/10.1145/1592761.1592779>
- Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children* (pp. 117–122). ACM. <https://doi.org/10.1145/1109540.1109556>
- Riyami, A. (2015). Main approaches to educational research. *International Journal of Innovation and Research in Educational Sciences*, 2(5), 412–416.



- Romero, M., Arnab, S., De Smet, C., Mohamad, F., Abdelouma, S., Minoi, J. L., et al (2018). Co-creativity assessment in the process of game creation. *European conference on games based learning* (pp. 549–XXI). Academic Conferences International Limited.
- Romero, M., Davidson, A.-L., Cucinelli, G., Ouellet, H., & Arthur, K. (2016). Learning to code: From procedural puzzle-based games to creative programming. *CIDUI proceedings. Learning and teaching innovation impacts*. ACUP.
- 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), 42. <https://doi.org/10.1186/s41239-017-0080-z>
- Rose, S., Habgood, J., & Jay, T. (2017). An exploration of the role of visual programming tools on the development of young children’s computational thinking. *Electronic Journal of E-Learning*, 15(4), 297–309. <https://doi.org/10.34190/ejel.15.4.2368>
- Rose, S. P., Habgood, M. J., & Jay, T. (2019). 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://doi.org/10.1145/3290607.3312871>
- Rovegno, I., & Dolly, J. P. (2006). Constructivist perspectives on learning. In D. Kirk, D. MacDonald, & M. O’Sullivan (Eds), *The Handbook of physical education* (pp. 242–261).
- Runco, M. A., & Jaeger, G. J. (2012). The standard definition of creativity. *Creativity Research Journal*, 24(1), 92–96. <https://doi.org/10.1080/10400419.2012.650092>
- Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59–66). ACM <https://doi.org/10.1145/2493394.2493403>
- Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. [https://eprints.soton.ac.uk/356481/1/Selby\\_Woollard\\_bg\\_soton\\_eprints.pdf](https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf)
- 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>
- Sivilotti, P. A., & Laugel, S. A. (2008). Scratching the surface of advanced topics in software engineering: A workshop module for middle school students. *ACM SIGCSE Bulletin*, 40(1), 291–295. <https://doi.org/10.1145/1352322.1352235>
- Sung, K., Hillyard, C., Angotti, R. L., Panitz, M. W., Goldstein, D. S., & Nordlinger, J. (2010). Game-themed programming assignment modules: A pathway for gradual integration of gaming context into existing introductory programming courses. *IEEE Transactions on Education*, 54(3), 416–427. <https://doi.org/10.1109/TE.2010.2064315>
- Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 148, 103798. <https://doi.org/10.1016/j.compedu.2019.103798>
- Tedre, M. (2017). *Many paths to computational thinking*. [Paper presentation] 3 TACCLE.
- Tekdal, M. (2021). Trends and development in research on computational thinking. *Education and Information Technologies*, 26(5), 6499–6529. <https://doi.org/10.1007/s10639-021-10617-w>

- Tikva, C., & Tambouris, E. (2020). *Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature review*. *Computers & Education*. <https://doi.org/10.1016/j.compedu.2020.104083>
- Ung, L. L., Labadin, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education*, 177, 104379. <https://doi.org/10.1016/j.compedu.2021.104379>
- Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, 1(2), 42–64. <https://doi.org/10.21623/1.1.2.4>
- Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via alice game-programming. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 427–432). ACM <https://doi.org/10.1145/2157136.2157263>
- Wilson, A., Connolly, T., Hailey, T., & Moffat, D. (2011). Evaluation of introducing programming to younger school children using a computer game making tool. In *Proceedings of the fifth European conference on games based learning* (pp. 639–649).
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725. <https://doi.org/10.1098/rsta.2008.0118>
- Wolz, U., Hallberg, C., & Taylor, B. (2011). *Scrape: A tool for visualising the code of Scratch programs*. [Poster presentation]. 42nd ACM Technical Symposium on Computer Science Education.
- Wolz, U., Maloney, J., & Pulimood, S. M. (2008). Scratch' your way to introductory CS. *ACM SIGCSE Bulletin*, 40(1), 298–299. <https://doi.org/10.1145/1352322.1352239>
- Wong, G. K. W., & Cheung, H. Y. (2018). 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>
- Zaharija, G., Mladenović, S., & Boljat, I. (2013). Introducing basic programming concepts to elementary school children. *Procedia - Social and Behavioral Sciences*, 106, 1576–1584. <https://doi.org/10.1016/j.sbspro.2013.12.178>
- Zhang, L.C., & 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>

## Authors Biographies

**Mohd Kamal Othman** is an Associate Professor in the Faculty of Cognitive Sciences and Human Development at Universiti Malaysia Sarawak. He received his PhD in Computer Science from the University of York UK, and his research interests span the fields of HCI, museum studies, tourism, digital technology and education.

**Syazni Jaslan** is a Digital Analytic trainee at Maybank concurrently pursuing MSc. in Data Science at Universiti Kebangsaan Malaysia. An enthusiast when it comes to

learning ever-changing methods on how to fill in the gaps between problems and solutions. She always believes that data-driven decisions play a fundamental role in progressing or breaking businesses.

**Fatin Afiqah Yamin** is a System Analyst at the biggest financial services in Malaysia. She has been involved in maintaining and analysing a system where it helps users identify insiders who are exercising market abuse. She portrays adaptability and open-mindedness throughout her involvements as a System Analyst. She believes by practising the values, she can build harmonies with others and progress as an individual. She received her BSc in Cognitive Science from Universiti Malaysia Sarawak.

**Shaziti Aman** is a Senior Lecturer whose research focuses on children, digital technology, museums and learning. She received her MA in Learning Sciences from Northwestern University.

**Fitri Suraya Mohamad** (ORCID ID 0000-0003-4460-8061) is an Associate Professor at the Department of Cognitive Sciences, University Malaysia Sarawak. Fitri Suraya does research in Pedagogic Theory, Higher Education Teaching and Learning and Educational Technology. She is currently working on gameful learning design for indigenous collaboration between communities in Sarawak.

**Nurfarahani Norman Anuar** is a UX Researcher at a tech company currently pursuing MSc. in Cognitive Sciences at Universiti Malaysia Sarawak (UNIMAS). She received her B.Sc. in the same field. She designed and developed a cultural heritage game-based learning application for children as part of her undergraduate program. She extends her interest in digital heritage to her postgraduate research.

**Abdulrazak Yahya Saleh** is a senior lecturer in the Faculty of Cognitive Sciences and Human Development at the Universiti Malaysia Sarawak. He received his PhD in Computer Science from the University Teknologi Malaysia (UTM), and his research interests focus on using Artificial Intelligence in various fields.

**Ahmad Azaini Abdul Manaf** (ORCID ID 0000-0003-0689-3949) is an Associate Professor in the faculty of creative Arts and Heritage at Universiti Malaysia Kelantan. He received his PhD in Service Design from Dongseo University, Busan, South Korea. His research interest is in the field of creative digital content, animation and game art and design education.