Does Everyone Use Computational Thinking?: A Case Study of Art and Computer Science Majors

Mr. Andreas Febrian, Utah State University

He received his bachelor and master degree in computer science (CS) from Universitas Indonesia, one of the top university in Indonesia. He was an active student who involved in various activities, such as research, teaching assistantship, and student organizations in the campus. He developed various CS skills through courses and research activities, especially in computer architecture, robotics, and web development. Through being a teaching assistant and joining student organizations, he developed an interest in psychology and Affective Computing. Currently, pursuing the Doctoral degree in Engineering Education at Utah State University with focuses in self-regulated learning in engineering design.

Dr. Oenardi Lawanto, Utah State University

Dr. Oenardi Lawanto is an associate professor in the Department of Engineering Education at Utah State University, USA. He received his B.S.E.E. from Iowa State University, his M.S.E.E. from the University of Dayton, and his Ph.D. from the University of Illinois at Urbana-Champaign. Before coming to Utah State, Dr. Lawanto taught and held several administrative positions at one large private university in Indonesia. He has developed and delivered numerous international workshops on student-centered learning and online learning-related topics during his service. Dr. Lawanto's research interests include cognition, learning, and instruction, and online learning.

Kamyn Peterson-Rucker
Alia Melvin
Mr. Shane E. Guymon

Does Everyone Use Computational Thinking? - A Case Study of Art and Computer Science Majors

Abstract

In this digital age, being computer literate and having computer science skills are essential, especially since most real-life solutions are technology-driven. Many K-12 and higher education institutions, states, and countries incorporate computational thinking (CT) into their curriculum. Although Wing describes CT as a problem-solving approach that utilizes fundamental computing concepts, which is applicable not only for scientists but everyone, most of the computational thinking instructional approaches are related to computer programming. Unfortunately, it is also unclear whether people use CT when solving non-programming problems. This study aims to answer two research questions: (1) In what ways do students use computational thinking skills when solving non-programming problems if any?; and (2) If students use CT when solving non-programming problems, in what ways do their approaches differ from computer science students? We conducted a qualitative multiple within-site case study research with three units of analysis. We recruited two students from computer science, a civil engineering student, an instructional design student, and an art student as cases, and asked them to think aloud while solving three problems. The collected think aloud data was transcribed and qualitatively coded to identify various CT skills. Our preliminary analysis of a computer science student and an art student reveals that they used various CT skills when solving all problems, and the application of CT skills was influenced by their background, experiences, and goals. Furthermore, we found that the art student was capable of utilizing various CT skills despite her lacked prior exposure to CS or CT, which shed new light on the nature of CT.

Introduction

In this computing era, various autonomous and semi-autonomous devices assist us in our home, work, and during travel [1], [2]. Some of these instruments can operate seamlessly, making us a step closer to achieve one of the digital age's visions that identified by Weisser (see [3]). Incredible as it is, most people believe this is not the peak of technological advancements and expect science and technology will continue to grow for an indefinite time. Nowadays, many businesses and industries prefer to utilize technology-integrated solutions when addressing problems, which then shaped the expected skill set of next-generation professionals [4], [5] and inspired numerous state legislators [6]–[10] to integrate computer science (CS) problem-solving approaches in their respective K-12 curriculums. As computational thinking (CT) courses become common in K-12 education [11]–[18], some higher education institutions also started offering computational thinking courses to students from any majors [4], [19]. At its heart, this movement reflects that computational thinking is everywhere and for everyone [20].

Various studies reported using text-based computer-programming- [13], [21]–[25], visual computer-programming [26]–[28], and puzzle instructional approaches [11], [29]–[31] to teach computational thinking. Most of these studies reported a positive impact on learning computing principles and an increase of interest in computer science in male and female students [12], [26], [28]. On the other hand, one study argues that exposing children to CT through various platforms is more effective in increasing their interest [32]. In 2016, McGill, Decker, and Settle reported that engaging in any computational activities during high school influences students' decision to pursue a CS degree, especially for female students [33].

Among the above reported instructional approaches for delivering computational thinking in K-12 and higher education[4], [11]–[13], [29], [21]–[28], [30], [31], most of them utilize computer programming. In these courses, the instructors are expecting students to change their perspective from software users to application creators [12], and encouragingly, some students live up to this expectation [34]. However, solely relying on programming-based instructions may limit students' ability to apply computational thinking skills in various situations, especially when solving non-computer programming-related problems, which is part of the computational thinking spirit [17], [26], [35]: in everywhere and for everyone. Still, it is also unclear whether these skills were used when dealing with non-programming problems. This exploratory case study aims to provide some insight on that issue.

Computational Thinking and Computer Science

Wing defines computational thinking as "an approach to solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing" (p.3717) [35]. The terms computing refers to computer science as a discipline, which is defined as a "systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application" (p.12) [36]. As a discipline, CS applies knowledge from mathematics (e.g., modeling the problem and solution), science (e.g., testing solution's accuracy and reliability), and engineering (e.g., developing usable systems) [36], [37]. Computer science concerns with both human-made information processes and human's cognitive enterprise [37].

Table 1. Computational thinking skills

Principle	Definition
abstraction and pattern generation	Identifying, populating, and organizing characteristics from an entity into a set of essential characteristics [35], [38].
systematic processing of information	A step-by-step agent-dependent instruction for processing a set of inputs into the desired unambiguous output, which is also known as algorithm [35], [37].
symbol systems and representations	Develop a model to store and express the characteristics and behaviors of an entity in an efficient way [37].
algorithmic notion of flow control	No precise definition found so far.
structured problem decomposition	Subdividing a computational problem into a simpler, more manageable subproblems [39]
iterative, recursive, and parallel thinking	Identifying, populating, and organizing a set of behaviors that can repeatedly be performed or at the same time [40].
conditional logic	Identify a set of criteria to allow or disregard the execution of an instruction set [40].
efficiency and performance constraint	Identifying potential efficiency and performance issues, and developing a method to enhance them [36]
debugging and systematic error detection	Evaluate and improve the program's accuracy, consistency, performance, and efficiency under various conditions [41], [42].

Although the current application of CS is pervasive, its core problem remains the same, which is developing and enhancing software systems. Similar to other design problems, a software design problem is typically an ambiguous, complex, and ill-structured problem that has multiple solutions and requires incorporation of knowledge from other domains [43] to satisfy client's needs and constraints [44]. In designing a software system, a computer scientist must represent the problem and solution as clearly as possible so that it can be efficiently executed by an information processing agent [17]. A computer scientists must also consider the solution's simplicity, accuracy, efficiency, usability, software and hardware reliability, robustness, evolvability, and security [36], [37], [42], [45].

After the former President Barack Obama wrote his first line of code during CS education week [46], many profit and nonprofit organizations joined the movement, for example, Google, which provides free computational thinking learning resources for students and instructors [47], [48]. Interestingly, there is no consensus on what comprised CT skills [14], [16], [49]–[51]. Nevertheless, Grover and Pea argue that most academicians agreed on nine computational thinking skills [17], and all are presented in Table 1 including the definitions. Unfortunately, we could not find a precise definition for 'algorithmic notion of flow control' yet, so we did not include it throughout the analysis process.

Purpose of Study

The purpose of this exploratory study is to provide insight on whether people use computational thinking skills when solving non-programming problems. We focus on assessing students' application of CT, and used these research questions as guidance:

1. In what ways do students use computational thinking skills when solving non-programming problems if any?
2. If students use computational thinking when solving non-programming problems, in what ways do their approaches differ from computer science students?

The Research Design

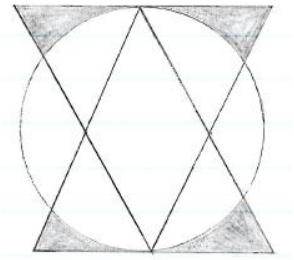
This study employs qualitative multiple within-site case study research design [52]. We use case study research design because this is an exploratory study, and there is limited literature on computational thinking in general problem-solving. This research is a multiple within-site case study because we recruited five students from our institution using convenient sampling method [53]. The bounded systems in this study are the students and their approach to solving given problems. There are three units of analysis for each case and are related to the first, second, and third problems.

We recruited five participants, which were two computer science majors, a civil engineering major, an instructional design major, and an art major. Creswell argues that a typical qualitative case study research involves five or fewer participants [52]. Therefore, we believe these cases are sufficient in number and diversity for this exploratory study. At the end of the data collection session, each participant received a $15 gift card as a token of appreciation.

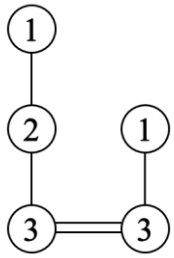
Qualitative Instrument

We developed three problems to assess the cases' problem-solving approaches. These problems have been face-validated, pilot-tested, and revised accordingly.

The first problem is inspired by one of the Computer Science Unplugged activities [54] that mimics the way a computer displays an image. The first problem consisted of two parts. In the first part, the participants were asked to provide verbal instructions for an amateur artist to redraw two patterned-images; please see Figure 1 (a) for a sample image. This problem is similar to computer programming because it requires the solver to provide a step-by-step instruction to be followed by an information-processing agent [17], [35], [39], [45], but instead of a computer, the agent is a human. This problem can be categorized as a design problem based on its similarity to programming [43], [55], [56]. When working on the first part, the artist could not see or talk to the participants and vice-versa, in a sense, two ways communication was prohibited. Thus, the participants must provide a detailed and precise step-by-step instruction so the artist can redraw the pattern as similar as possible. After describing an image, the participants were allowed to see the artist's result and adjust their approach when providing instruction for the next pattern. In the second part, we gave the participants a hypothetical question where they were asked to describe 4 and 25 images in a limited time. The goal of this question was to assess whether they would use different strategies when the number of images increased rapidly.



(a) An image example of the first problem



(b) A puzzle example for the second problem

Figure 1. Problem examples.

The second problem contains two bridge-puzzles, one easy and the other of medium difficulty; please see Figure 1 (b) for example. The puzzle rules are (a) to connect each island, which is represented as a circle, with the number of bridges shown inside the circle; (b) there can only be two bridges connecting two islands; (c) a bridge must not overlap with other bridges or islands; and (d) there must be continuous link between all islands, which means there cannot be isolated island. Figure 1 (b) presents a solution that satisfies the puzzle's rules.

When solving the second problem, the participants worked on a computer because we used existing online bridge-puzzles from http://www.nikoli.com/en/puzzles/hashiwokakero/. The Nikoli website provides ten sample puzzles; all are written in Adobe action scripts. The puzzle application also has a button for verifying user's solution. Since the participants answer both puzzles on a computer, we also recorded the computer screen.

The bridge-puzzle is an example of a constraint satisfaction problem [57], [58], in which the problem required the solver to reach the solution state without compromising any given rules. Thus, this problem can be considered as a rule-using problem [43], [55], [56]. We expect the

computer scientists would excel in solving the second problem because constraint satisfaction problems are common problems in computer science. Furthermore, puzzles are also commonly used in CS instructional approaches [11], [29]–[31], [59], [60].

The third problem was a shopping task, where the participants received a hypothetical grocery store floor plan and a shopping list. Using the given floor plan, the participants had to describe their route for acquiring all items in the list including a gallon of milk, a jar of jam, a five-pound bag of flour, five pounds of chicken, and a bottle of fruit juice. No specific constraint was given to the participants. At the end of this task, the participants were asked to describe their usual shopping strategy.

We believe people develop a general floor plan of their favorite stores, which was confirmed during the analysis. A hypothetical grocery store floor plan was given to the participants so we can correctly compare the participants' shopping approaches. This problem can be considered as a decision-making problem [43], [55], [56].

Data Collection

Three researchers were present in each data collection session, where one researcher interviewed the participant, one acted as an amateur artist for the first problem, and the last one monitored various recording tools. The data collection was conducted in one of the laboratories at our institution. This spacious lab was selected because it provided a quiet environment and enabled us to have an adequate distance between the participants and the artist. The participants were asked to use verbal protocol so that we could assess their thinking process. Although the verbal protocol is widely used in research [61], sometimes the participants forget to speak or get more cognizant on their thought process which then helps them to think more deeply about the problem [62]. To minimize chances of the participants forgetting to use verbal protocol, we conducted a practice session to make them familiar with the protocol. Additionally, we prepared prompts for reminding them to think aloud.

Data Analysis

The analysis process started with transcribing all recorded data. Two researchers worked on the transcriptions, and we assumed it was conducted with minimal error. The transcriptions served as a method to reduce data complexity so it can be analyzed easier [63]. We then qualitatively coded the transcription based on Table 1 to identify participants' CT skills and the contexts surrounding those identified activities. We postulated some CT skills could not be identified explicitly but could be inferred from its nature. Our analysis process confirmed that supposition, for example, when describing the first image, one participant said:

> "The square will be sectioned off into four triangles after those three previous steps. The triangles are on the left and right, not the top and bottom; they should be lined, like, would that be hatching? I think hatching is several different sets of lines; I do not know. You need to horizontally strip the right and left triangles inside the square." - Amara

Amara's verbalization exhibited 'structured problem decomposition' and 'systematic processing of information' skills in which she tried to provide a clear and unambiguous step-by-step instruction. However, this example also displayed her ability to use 'abstraction and pattern

generation' skills because she was able to identify various patterns which guided her in decomposing the image. Using the coded transcriptions, we will describe the nature of participants' problem-solving approach and answered the research questions.

Current Progress

We have recruited and collected data from five participants, who voluntarily consented to participate in this study after hearing the research purpose and procedure, and selected aliases to protect their identity. We have transcribed all participants' recorded data and coded two transcriptions. Some of our preliminary findings on these two transcriptions are discussed in this paper.

Participants

The first participant was Amara, an eighteen year old female undergraduate student who was majoring in art with an emphasis in drawing and never learned any computer programming. She identified herself as a mix of white Caucasian and African American. Her favorite learning style is kinesthetic, followed by visual and auditory learning style.

The second participant was BeMyGuest, a forty-four year old male graduate student who was majoring in computer science. He identified himself as a Hispanic that preferred the kinesthetic learning style, followed by visual and auditory learning style.

Preliminary Findings and Discussion

As presented in Table 2, all participants were observed using the 'abstraction and pattern generation,' 'systematic processing of information,' 'symbol systems and representations,' and 'structured problem decomposition' skills for all problems; from this point forward, we will use prevalent-CT skills to refer to the above-listed skills. Naturally, some differences were also found among the participants' approach for each problem, which we believe was due to the diversity of their traits, knowledge, experiences, and personal goals. For examples, Amara mentioned that she has a limited short-term memory and does not like to lift heavy things, and BeMyGuest remarked that he is a family-man and sometimes faces difficulty in finding appropriate words in English.

The First Problem

As an art major, Amara had many experiences related to the first problem. When working on the first part, she tried to assess the artist's knowledge, skills, and ability to communicate. Some of her assessments were so detailed, for example, she said, "I assume you know what a square is?" Unfortunately, the artist was prohibited from answering any questions. Amara elaborated the importance of such assessment by saying, "In my general experience when you are trying to get someone to draw an exact copy of a specific curve it varies depending on the artist's skill and how you describe it." Thus we believe that she was planning to adjust her instruction to match the artist's knowledge and skills.

When providing the instruction, Amara always started with the basic shapes. She said, "When you are drawing, you have to start with the basic shapes, and the most basic and easiest shape to understand in this particular drawing is the square." Starting out with a basic shape was very important for her because it helped in laying out the entire drawing correctly; she

mentioned, “You could probably get the right shape by drawing a cross; you will have to erase it later, but it would get the basic layout.” We believe her instructions were the product of utilizing the prevalent CT skills.

Table 2. Utilization summary of participants’ computational thinking skills on each problem

CT Skill	Amara’s CT on			BeMyGuest’s CT on		
	#1	#2	#3	#1	#2	#3
abstraction and pattern generation	x	x	x	x	x	x
systematic processing of information	x	x	x	x	x	x
symbol systems and representations	x	x	x	x	x	x
structured problem decomposition	x	x	x	x	x	x
iterative, recursive, and parallel thinking	x	x		x	x	
conditional logic		x			x	
efficiency and performance constraint	x		x		x	
debugging and systematic error detection	x			x		

Amara also sometimes exploited repeating patterns, for example, “Once you have drawn that on each prong, on the inside of the first petal you drew, about halfway down the cross, do it again.” She also utilized the ‘debugging and systematic error detection’ skill when dealing with analogies; she said, “It is shaped like a lotus, except it does not have enough leaves. Perhaps, they are more like petals?” Furthermore, she was observed using the ‘efficiency and performance constraint’ skill by trying to shorten her instruction. She explained, “Because I know for a fact that if you try to describe how to draw something to someone and you use too many words, they will get confused very easily.”

When working on the second part of this problem (i.e., describing multiple pictures), Amara used different approaches because the goal of the first and second part is different; she said:

> “When you are describing something so someone can draw it, you are not describing it in its entirety; you are describing the steps it takes to create the image. Describing an image without the intent of drawing is completely a different thing.”

Thus, by developing an accurate interpretation of a problem, Amara was able to choose the most appropriate approach.

As a computer scientist, BeMyGuest might not have a lot of practice in providing drawing instructions, but he had extensive experience in programming. When working on the first part, he assessed the artist’s ability to communicate, which we believe, would allow him to handle this problem as a debugging process. When debugging, a computer programmer develops familiarity with the assumed inputs, information processing method, and outputs of the program, and based on that familiarity, an experienced programmer identifies problematic code segments [64], [65]. Similarly, the artist’s communication ability would allow BeMyGuest to develop familiarity and identify potential issues.

When providing instructions, he did not mention any specific approach like Amara. However, we observed that he started by describing the general image shape, for example, “This is like a geometry figure, a hexagon” and “This looks like a flower with many leaves, with four leaves.” We believe his instructions were the product of utilizing the prevalent CT skills. He was observed using the ‘iterative, recursive, and parallel thinking’ skill; he said, “So take one triangle, go from the edge to the center, and draw as many parallel lines as you can to fill the triangle.” Additionally, BeMyGuest was observed employing the ‘debugging and systematic error detection’ skill when he was correcting himself using the word leaves instead of petals.

BeMyGuest also tried to keep his instruction short, but he did not consider it as a performance constraint. We believe that strategies are most effective when being used consciously [66][67]. Thus, we also did not consider his short instruction as a performance constraint. After providing instruction for the first image, BeMyGuest reported that he was not satisfied with his instruction. He said, “I am 8% satisfied with my description because I was not able to describe these things and the curvy branches.”

When working with the second part, BeMyGuest did not change his approaches. Based on his descriptions, he assumed that the artist still would have to draw the images by following his instructions; he said, “Then I would try to make the other person draw the figure that is the base and then the pyramid that is over here.” We believe his assumption was because English was his second language.

The Second Problem

Amara started by reading and understanding the puzzle’s rules. She then began solving the first puzzle while developing a familiarity with the application. Her approach was to complete the islands with the biggest number first, and then continue to the second largest. We believe her approaches to solving the problem were the products of utilizing the prevalent CT skills. Furthermore, she repeated some of her approaches on several islands, which then we considered as the application of the ‘iterative, recursive, and parallel thinking’ skill. Amara was also observed using the ‘conditional logic’ skill, for example, Amara incorporated surrounding islands’ states to determine her next move; she said, “So if it has two possible bridges, which means this one maybe not here. However, this number five has to be tied to an island, so the only option for this.” When applying this skill, Amara tested her assumptions so she can move forward. In the end, she was able to solve the first puzzle but failed to solve the second because she was heavily depended on the trial-and-error approach.

We believe the second problem was more natural for BeMyGuest because puzzles are commonly used in CS instructional approaches [11], [29]–[31], [59], [60]. BeMyGuest’s familiarity was apparent after he successfully solved the first puzzle, in which he deemed himself taking too much time to solve it. Such feeling implied that time was one of his success criteria in this problem, which then we considered as his ‘efficiency and performance constraint.’ He managed to solve the second puzzle faster, which indicates that he learned something from solving the first puzzle.

Similar to Amara, he started by reading and understanding the puzzle rules, and then developing familiarity with the application. His initial approach was to deal with the biggest and smallest number (i.e., eight and one respectively). He also repeatedly using the same approach to

numerous island, for example, “I am trying to get the eight-numbers for this one, so all the eight-links are done.” He was also observed using ‘conditional logic.’ BeMyGuest reported that he changed his approach when solving the second puzzle. He reported that he tried to minimize utilizing the trial-and-error approach and that he also considered the significance of even numbers; he said, “The main thing that changed was the even numbers. So I tried to complete the 8s, then 6s, then trying to see what are the 4s doing.”

The Third Problem

In the third problem, aside from the prevalent-CT skills, Amara reported using the ‘efficiency and performance constraint’ skill. After checking the shopping list, Amara decided that time was her constraint and she said, “I would want this [shopping task] to go as quick as possible, and I would want to get the heavier items at the end.” On the other hand, BeMyGuest did not determine any specific constraint and just cruised along the aisles to get the items.

Both participants reported that their shopping experience would change depending on their goals and budget. If they have no specific goal or budget, then they will take their time; as BeMyGuest said, “I am a family man, so I like to go shopping with my family and try to go everywhere when I have money. If I do not have money, I will directly get the items and then leave.”

Limitations

The first limitation is related to the design of this study, which is the qualitative multiple within-site case study research design. Since case study research typically does not involve many participants [52], including this study, the findings are not generalizable. However, that does not mean this study has no benefit. Teague argues that CS can always benefit from detailed descriptions of students, perception, and attitudes in learning and applying various computing principles [68]. Furthermore, academic scholars start to understand that findings from educational studies that involves a huge number of participants cannot be applied broadly to students [66]. Thus this study enriches our understanding and relatedness on the nature and application of computational thinking in various situations. The second limitation is related to participant diversity that our study did not assess people’s computational thinking skills from all disciplines and profession. Furthermore, our study did not assess people CT skills for all types of problems in varying contexts. The third limitation is we do not assess participants’ prior exposure to computer science and computational thinking thoroughly.

Preliminary Conclusion and Implication

Our preliminary analysis suggests computer scientists and artists used computational thinking skills when dealing with various problems. The preliminary answer to the first question was provided in the Preliminary Findings and Discussion section. In summary, we observed that our participants used ‘abstraction and pattern generation,’ ‘systematic processing of information,’ ‘symbol systems and representations,’ and ‘structured problem decomposition’ skills and that the application of those skills was adjusted based on the problem. We also found that the problem characteristics also affect what other computational thinking skills are used by the participants, for example, we observed that all participants used the ‘conditional logic’ skill when solving the second problem but not in the first and third problems. Furthermore, the solver’s characteristics influence the computational thinking skills used in solving the problem.

For example, when dealing with the shopping problem, Amara was observed using ‘efficiency and performance constraint’ due to her preference for not lifting heavy items, and we did not observe any constraints used by BeMyGuest.

The preliminary answer to the second question is that students’ background knowledge and experience influence their approaches, including using CT skills. The differences between Amara’s and BeMyGuest’s approaches when solving the first and second problem were evidence of this. However, there is a possibility that when dealing with daily problems, such as shopping, no fundamental differences can be observed between CS and non-CS students. Therefore, knowing computational thinking skills does not imply the utilization of those skills. Some researchers argue that students’ learned actions will be bounded to the contexts surrounding that learning [69], [70]. Thus, applying known skills in a certain context to another context requires a competence in learning transfer, which is also a skill in itself.

The fact that Amara utilized various CT skills when solving problems suggests that some computational thinking skills are applicable in various contexts, as claimed by many [17], [26], [35]. We argue that CT is a problem-solving approach [71], and like other techniques, its applicability constrained by contexts surrounding a problem and the solver. Furthermore, the fact that Amara had no experience in computer science suggests that one does not have to be a computer scientists to acquire computational thinking skills. Therefore, exposing students to programming and various CS problems might not be the only way to teach computational thinking. Lye and Koh argue that helping students to reflect on their learning experience is an essential step for acquiring CT skills [15]. Perhaps helping students to be reflective and more aware of their thinking process could be used as an alternative approach, such as teaching them various metacognition and self-regulation strategies [72]–[74]. After all, CS also draws knowledge from human’s cognitive process [37].

In regards to education, we need to reexamine our purpose and method of teaching CT skills. It is true that technology has become an essential part of our daily life and employers prefer to use technology-driven solutions. However, one does not need to know computer programming to be literate in using a computer or other software applications.

Future Task

We will focus further endeavors on analyzing data from the rest of the participants (i.e., third, fourth, and fifth), and then integrate all the analysis results.

Bibliography

[1] A. Bundy, “Computational Thinking is Pervasive,” *J. Sci. Pract. Comput.*, vol. 1, no. 2, pp. 67–69, 2007.

[2] Q. Bui, “Will Your Job Be Done By A Machine?,” *Planet Money - The Economy Explained*, 2015. [Online]. Available: http://www.npr.org/sections/money/2015/05/21/408234543/will-your-job-be-done-by-a-machine. [Accessed: 25-May-2015].

[3] M. Weisser, “The Computer for the Twenty-First Century,” *Sci. Am.*, vol. 3, no. 265, pp. 94–104, 1991.

[4] S. Hambrusch, C. Hoffmann, J. T. Korb, M. Haugan, and A. L. Hosking, “A Multidisciplinary Approach Towards Computational Thinking for Science Majors,” *ACM SIGCSE Bull.*, vol. 41, no. 1, p. 183, Mar. 2009.

[5] P. B. Henderson, “Ubiquitous computational thinking,” *Computer (Long. Beach. Calif).*, vol. 42, no. 10, pp. 100–102, Oct. 2009.

[6] M. Chaudhry, “Your Kids Aren’t Robots, And That’s Exactly Why They Must Know How To Code,” *Forbes*, Washington, DC, 26-Aug-2015.

[7] J. Carpenter, “Chicago private schools lead ‘high-tech, high-touch’ movement,” *Chicago Tribune*, Chicago, 28-Aug-2015.

[8] A. O. Stallings, *S.B. 107 Computer Science Initiative for Public Schools (Filed)*. 2015, p. S.B. 107.

[9] E. Kao, “Exploring Computational Thinking,” *Google Research Blog*, 2010. [Online]. Available: http://googleresearch.blogspot.com/2010/10/exploring-computational-thinking.html. [Accessed: 28-Aug-2015].

[10] K. Wilson, “STEM in K-5: Start computational thinking early!,” International Society for Technology in Education, United States, 2014.

[11] D. Isayama, M. Ishiyama, R. Relator, and K. Yamazaki, “Computer Science Education for Primary and Lower Secondary School Students,” *ACM Trans. Comput. Educ.*, vol. 17, no. 1, pp. 1–28, Sep. 2016.

[12] I. Lee, F. Martin, and K. Apone, “Integrating computational thinking across the K--8 curriculum,” *ACM Inroads*, vol. 5, no. 4, pp. 64–71, Dec. 2014.

[13] I. Fronza, N. El Ioini, and L. Corral, “Teaching Computational Thinking Using Agile Software Engineering Methods: A Framework for Middle Schools,” *ACM Trans. Comput. Educ.*, vol. 17, no. 4, pp. 1–28, Aug. 2017.

[14] A. Yadav, H. Hong, and C. Stephenson, “Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms,” *TechTrends*, vol. 60, no. 6, pp. 565–568, Nov. 2016.

[15] S. Y. Lye and J. H. L. Koh, “Review on teaching and learning of computational thinking through programming: What is next for K-12?,” *Comput. Human Behav.*, vol. 41, pp. 51–61, Dec. 2014.

[16] N. Senske, “A Curriculum for Integrating Computational Thinking,” in *ACADIA Regional 2011 Parametricism*, 2011, pp. 91–98.

[17] S. Grover and R. Pea, “Computational Thinking in K−12 : A Review of the State of the Field,” *Educ. Res.*, vol. 42, no. 1, pp. 38–43, Feb. 2013.

[18] J. V. Ernst and A. C. Clark, “Fundamental Computer Science Conceptual Understandings for High School Students Using Original Computer Game Design,” *J. STEM Educ. Innov. Res.*, vol. 13, no. 5, pp. 40–45, 2012.

[19] B. C. Czerkawski and E. W. Lyman, “Exploring Issues About Computational Thinking in Higher Education,” *TechTrends*, vol. 59, no. 2, pp. 57–65, Jan. 2015.

[20] J. M. Wing, “Computational Thinking,” *Commun. ACM*, vol. 49, no. 3, p. 33, 2006.

[21] J. C. Adams, “Alice, middle schoolers & the imaginary worlds camps,” *ACM SIGCSE Bull.*, vol. 39, no. 1, p. 307, Mar. 2007.

[22] A. Ruf, A. Mühling, and P. Hubwieser, “Scratch vs. Karel: impact on learning outcomes and motivation,” in *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE ’14*, 2014, pp. 50–59.

[23] J. Cao, S. D. Fleming, M. Burnett, and C. Scaffidi, “Idea Garden: Situated Support for Problem Solving by End-User Programmers,” *Interact. Comput.*, vol. 27, no. 6, pp. 640–660, Nov. 2015.

[24] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver, “Scratch: Programming for All,” *Commun. ACM*, vol. 52, no. 11, p. 60, Nov. 2009.

[25] B. Aynsley, “Coding as part of school curriculum a matter of time and resources,” *The Australian*, Sydney, 25-Aug-2015.

[26] E. B. Witherspoon, R. M. Higashi, C. D. Schunn, E. C. Baehr, and R. Shoop, “Developing Computational Thinking through a Virtual Robotics Programming Curriculum,” *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–20, Oct. 2017.

[27] D. Weintrop and U. Wilensky, “Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms,” *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–25, Oct. 2017.

[28] A. Theodoropoulos, A. Antoniou, and G. Lepouras, “How Do Different Cognitive Styles Affect Learning Programming? Insights from a Game-Based Approach in Greek Schools,” *ACM Trans. Comput. Educ.*, vol. 17, no. 1, pp. 1–25, Sep. 2016.

[29] J. Krauss, *Computer Science-in-a-Box: Unplug Your Curriculum*. Boulder, CO: The National Center for Women & Information Technology, 2008.

[30] A. S. Marzocchi, “Using the Tower of Hanoi puzzle to infuse your mathematics classroom with computer science concepts,” *Int. J. Math. Educ. Sci. Technol.*, vol. 47, no. 5, pp. 814–821, Jul. 2016.

[31] F. Jordan, “CS Unplugged - Creating a Butter and Jam Sandwich,” 2014. [Online]. Available: https://www.youtube.com/watch?v=6iPfSIxrP18. [Accessed: 28-Mar-2015].

[32] A. Merkouris, K. Chorianopoulos, and A. Kameas, “Teaching Programming in Secondary Education Through Embodied Computing Platforms,” *ACM Trans. Comput. Educ.*, vol. 17, no. 2, pp. 1–22, May 2017.

[33] M. M. McGill, A. Decker, and A. Settle, “Undergraduate Students’ Perceptions of the Impact of Pre-College Computing Activities on Choices of Major,” *ACM Trans. Comput. Educ.*, vol. 16, no. 4, pp. 1–33, Jun. 2016.

[34] D. A. Fields, Y. B. Kafai, and M. T. Giang, “Youth Computational Participation in the Wild: Understanding Experience and Equity in Participating and Programming in the Online Scratch Community,” *ACM Trans. Comput. Educ.*, vol. 17, no. 3, pp. 1–22, Aug. 2017.

[35] J. M. Wing, “Computational Thinking and Thinking About Computing,” *Philos. Trans. A. Math. Phys. Eng. Sci.*, vol. 366, no. 1881, pp. 3717–25, Oct. 2008.

[36] P. J. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young, “Computing as a Discipline,” *Commun. ACM*, vol. 32, no. 1, pp. 9–23, Feb. 1989.

[37] P. J. Denning, “Computer Science,” in *Encyclopedia of Computer Science*, 4th ed., A. Ralston, E. D. Reilly, and D. Hemmendinger, Eds. Chichester, UK: John Wiley and Sons Ltd., 2003, pp. 405–419.

[38] TechTarget, “What Is.” [Online]. Available: http://whatis.techtarget.com. [Accessed: 24-Apr-2017].

[39] K. D. Lee, *Foundations of Programming Languages*. Cham: Springer International Publishing, 2014.

[40] Computer Hope, “Free Computer Help and Information.” [Online]. Available: http://www.computerhope.com. [Accessed: 22-Apr-2017].

[41] P. J. Denning and P. A. Freeman, “The Profession of IT Computing’s Paradigm,” *Commun. ACM*, vol. 52, no. 12, p. 28, Dec. 2009.

[42] P. J. Denning, “Great principles in computing curricula,” *ACM SIGCSE Bull.*, vol. 36, no. 1, pp. 336–341, Mar. 2004.

[43] D. H. Jonassen, “Toward a design theory of problem solving,” *Educ. Technol. Res. Dev.*, vol. 48, no. 4, pp. 63–85, Dec. 2000.

[44] Engineering Accreditation Commission, “Criteria for accrediting engineering program,” *ABET Report E1 11/19*, vol. 3. Baltimore, Md, 2003.

[45] M. Clark, “Computer Science: A hard-applied discipline?,” *Teach. High. Educ.*, vol. 8, no. 1, pp. 71–87, Jan. 2003.

[46] E. Mechaber, “President Obama Is the First President to Write a Line of Code,” *White*

House Website, 2014. [Online]. Available: https://www.whitehouse.gov/blog/2014/12/10/president-obama-first-president-write-line-code. [Accessed: 30-Aug-2015].

[47] Google Inc., "Computational Thinking for Educators - Course." [Online]. Available: https://computationalthinkingcourse.withgoogle.com.

[48] Google, "Google: Exploring Computational Thinking." [Online]. Available: http://www.google.com/edu/computational-thinking/. [Accessed: 28-Mar-2015].

[49] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," *... 2012 Annu. Meet. ...*, 2012.

[50] J. Moreno-León, G. Robles, and M. Román-González, "Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking," *RED. Rev. Educ. a Distancia*, vol. 15, no. 46, 2015.

[51] The CSTA Standards Task Force, *CSTA K-12 Computer Science Standards*, Revised 20. New York, New York, USA: ACM, 2011.

[52] J. W. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*, 3rd ed. SAGE Publications, 2012.

[53] I. T. Coyne, "Sampling in qualitative research. Purposeful and theoretical sampling; merging or clear boundaries?," *J. Adv. Nurs.*, vol. 26, no. 3, pp. 623–630, Sep. 1997.

[54] CS Education Research Group, "Computer Science Unplugged." [Online]. Available: http://csunplugged.org/. [Accessed: 28-Mar-2015].

[55] D. H. Jonassen, *Learning to solve problems: A handbook for designing problem-solving learning environments*. Routledge, 2010.

[56] D. H. Jonassen, *Learning to Solve Problems: An Instructional Design Guide*. John Wiley & Sons, 2004.

[57] D. Andersson, "Hashiwokakero is NP-complete," *Inf. Process. Lett.*, vol. 109, no. 19, pp. 1145–1146, Sep. 2009.

[58] R. F. Malik, R. Efendi, and E. A. Pratiwi, "Solving Hashiwokakero Puzzle Game with Hashi Solving Techniques and Depth First Search," *Bull. Electr. Eng. Informatics*, vol. 1, no. 1, 2012.

[59] H. Geffner, "Artificial Intelligence: From programs to solvers," *AI Commun.*, vol. 27, no. 1, pp. 45–51, 2014.

[60] R. Jeffries, P. G. Polson, L. Razran, and M. E. Atwood, "A process model for Missionaries-Cannibals and other river-crossing problems," *Cogn. Psychol.*, vol. 9, no. 4, pp. 412–440, Oct. 1977.

[61] L. Bainbridge and P. Sanderson, “Verbal Protocol Analysis,” in *Evaluation of Human Work, 3rd Edition*, 3rd ed., J. R. Wilsom and N. Corlett, Eds. CRC Press, 2005, p. 1048.

[62] M. T. H. Chi, N. De Leeuw, M.-H. Chiu, and C. Lavancher, “Eliciting Self-Explanations Improves Understanding,” *Cogn. Sci.*, vol. 18, no. 3, pp. 439–477, 1994.

[63] J. C. Lapadat and A. C. Lindsay, “Transcription in Research and Practice: From Standardization of Technique to Interpretive Positionings,” *Qual. Inq.*, vol. 5, no. 1, pp. 64–86, Mar. 1999.

[64] M. Weiser, “Programmers use slices when debugging,” *Commun. ACM*, vol. 25, no. 7, pp. 446–452, Jul. 1982.

[65] Y.-T. Lin, C.-C. Wu, T.-Y. Hou, Y.-C. Lin, F.-Y. Yang, and C.-H. Chang, “Tracking Students’ Cognitive Processes During Program Debugging—An Eye-Movement Approach,” *IEEE Trans. Educ.*, vol. 59, no. 3, pp. 175–186, Aug. 2016.

[66] W. Lake, W. Boyd, and W. Boyd, “Understanding How Students Study: The Genealogy and Conceptual Basis of A Widely Used Pedagogical Research Tool, Biggs’ Study Process Questionnaire,” *Int. Educ. Stud.*, vol. 10, no. 5, p. 100, Apr. 2017.

[67] J. B. Biggs, *Study Process Questionnaire Manual*. Melbourne, Australia: Australian Council for Educational Research Ltd., 1987.

[68] D. Teague, “A People-First Approach to Programming,” in *ACE ’09 Proceedings of the Eleventh Australasian Conference on Computing Education*, 2009, pp. 171–180.

[69] L. Reder, J. R. Anderson, and H. A. Simon, “Situated Learning and Education,” *Educ. Res.*, vol. 25, no. 4, pp. 5–11, 1996.

[70] C. Bereiter, “Situated Cognition and How to Overcome It,” in *Situated Cognition: Social, Semiotic and Psychological Perspectives*, D. Kirshner and J. A. Whitson, Eds. Hillsdale, NJ: Erlbaum, 1997, pp. 281–300.

[71] R. L. Glass, “Call It Problem Solving, Not Computational Thinking,” *Communications of the ACM*, vol. 49, no. 9, p. 3, 2006.

[72] J. H. Flavell, “Metacognitive Aspects of Problem Solving,” in *The Nature of Intelligence*, L. B. Resnick, Ed. Hillsdale, NJ, USA: Erlbaum, 1976, pp. 21–64.

[73] D. L. Butler, L. Schnellert, and N. E. Perry, *Developing Self-Regulating Learners*. Toronto, ON, Canada: Pearson Education Inc., 2017.

[74] S. P. Lajoie, “Metacognition, Self Regulation, and Self-regulated Learning: A Rose by any other Name?,” *Educ. Psychol. Rev.*, vol. 20, no. 4, pp. 469–475, Sep. 2008.