INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN PROGRESSIVE EDUCATION & DEVELOPMENT

OPEN ACCESS JOURNAL

Vol 12, Issue 1, (2023) E-ISSN: 2226-6348

Word Problems as a Vehicle for Teaching Computational Thinking

Ku Soh Ting
Department of Science and Technical Education, Faculty of Educational Studies, 43400 Universiti Putra Malaysia, Serdang, Selangor

Othman Talib
Department of Science and Technical Education, Faculty of Educational Studies, 43400 Universiti Putra Malaysia, Serdang, Selangor

Ahmad Fauzi Mohd Ayub
Department of Foundations of Education, Faculty of Educational Studies 43400 Universiti Putra Malaysia, Serdang, Selangor

Maslina Zolkepli
Department of Computer Science, Faculty of Computer Science & Information Technology 43400 Universiti Putra Malaysia, Serdang, Selangor

Chen Chuei Yee
Department of Mathematics, Faculty of Science 43400 Universiti Putra Malaysia, Serdang, Selangor

Teh Chin Hoong
ASASIpintar Program, Pusat GENIUS@Pintar Negara 43600 Universiti Kebangsaan Malaysia, Bangi, Selangor

To Link this Article: http://dx.doi.org/10.6007/IJARPED/v12-i1/16543 DOI:10.6007/IJARPED/v12-i1/16543

Published Online: 23 March 2023

Abstract

The 4th Industrial Revolution is sweeping the world, incorporating technology into communities. High-tech tools and resources are being developed as part of the digital revolution. In the realm of education, there has been a larger emphasis on coding instruction in order to develop a sufficient number of young people to fill fifty percent of computing-related STEM (science, technology, engineering, and mathematics) job openings. Our children and grandchildren must think critically in order to solve the world's ill-structured, unexpected, and intricate issues. Computational thinking, when combined with critical thinking, can produce automatic or semi-automated problem-solving solutions. As a result, computational thinking is becoming increasingly important in science, mathematics, and nearly every other

topic. This is demonstrated by Malaysia's recent implementation of computer coding into the school curriculum in order to build 21st century competencies in students. Computational thinking (CT) is one of the conceptual underpinnings required to solve problems successfully and efficiently by combining facts and ideas. In order to solve problems and offer solutions that can be applied in a variety of contexts, computational thinking—i.e., algorithmically, with or without the use of computers—is crucial in computing and information science. It is critical to foster a diversity of abilities and competences among students as to successfully navigate the complexity challenge in the real world. As a result, building a learning environment can be a daunting task for many instructors who lack the tools and research-based expertise to redesign their teaching methods. This conceptual framework has two objectives: (1) developing deep learning and connected computational thinking through a mathematical curriculum instructional model which is capable of improving students' problem-solving skills; and (2) implementing the designed model to assist teachers in education who encountered difficulties when using problem-based curriculum materials. A computational mathematics problem-based learning (CM-PBL) teaching technique is developed to achieve these goals. Students can use the CM-PBL learning framework to simulate and build their own computational models as to aid their self-learning and comprehension of mathematical concepts. It demonstrates that how students can use coding to improve their soft engineering skills.
Keywords: STEM, Coding, Problem Solving, 21st Century Skills

Introduction

Computing facilitates and drives the integration of several technologies in today's society. Barr and Stephenson (2011) argued that computing principles nowadays, influence every aspect of a student's life and work, from utilizing loyalty cards to do scientific research. Majority of students today were born and reared with knowledge of internet technologies. Learners spent their entire lives interacting with digital music players, videogames, mobile devices, and other digital-age technologies. Learners equipped with smartphones and laptops that are becoming more sophisticated. However, how many pupils are truly understanding how computers, software, games, and mobile applications work? As a result, students can learn basic coding to understand how these technological devices work.

Why integrate computer coding in education?

Jobs in science, technology, engineering, and mathematics (stem) have increased fast in this century, particularly in the computer field. Stem-related jobs are rising three times faster than non-stem employment, while 2.4 million jobs are unfilled. Computer and math jobs account for half of all stem jobs. Malaysia's top five digital talents, according to linkedin's digital workforce of the future research, are digital. Stem-related jobs stimulate innovation and are seen as the most desirable jobs of the future (Langdon et al., 2011).

Coding skills are now regarded as an essential and play important roles in stem industries. The most significant aspect of computing and the sharing of ideas for the growth of ct is coding. There has been an increase in the early introduction of computational coding skills as the necessity of computer coding skills has been recognised until the high education (Allan et al., 2010) which involved the combination of this competency with other essential competencies such as reading, writing, and mathematics. It is believed that the utilisation of a wide variety of coding tools during early curricular learning can influence the degree to which "computer-like behaviour" develops. Computational modelling is a useful method for understanding difficult mathematical concepts (Hambrusch et al., 2009) because it closely aligns mathematics closely with computer programming and thus brings mathematics to life (Felleisen & Krishnamurthi, 2009).

Many countries are rapidly changing their elementary and secondary school curricula to include computational thinking (CT) as a component of their 21st-century skills. In recent years, these countries have incorporated computer coding into their educational curricula which including the UK (England), Finland, Belgium, Czech Republic, Malta, France, Austria, Bulgaria, Hungary, Denmark, Estonia, Lithuania, Ireland, Israel, Poland, Spain, Slovakia, Portugal and so on. To develop a scientific and innovative society, one of the objectives of our vision 2020 is to produce a sufficient number of qualified STEM graduates (Ministry of Education Malaysia, 2013). According to STEM Initiative in Malaysia Education Blueprint 2013-2025, National and school-based evaluations emphasize creativity and problem-solving abilities (Ministry of Education Malaysia, 2013). In addition to fostering CT, logical reasoning, and critical thinking, computer coding helps pupils in comprehending 21st century reality. According to research studies, computer education and computational thinking have the potential to develop students' problem-solving abilities, higher-order thinking abilities, communication skills, and collaboration skills in ways that promote learning across the curriculum and empower students to become innovative technology creators. These skills can be cultivated in methods that foster cross-curricular learning and empower students to become inventive technology developers. Reading, writing, and mathematics are currently taught as core topics in schools, which is suitable because these subjects provide benefits that may be applied in a variety of fields. "Learning to write code broadens your mind and helps you think better; it develops a technique of thinking that I believe is valuable in all industries," Bill Gates was quoted as saying. Furthermore, this technique ensures that all students have access to computing training and can link with core curriculum in novel and innovative ways. Wing (2006) advocated the necessity of computational thinking in our education by stating, “To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability”.

Do Learners Really Need Computer Thinking?

“Computational thinking is a fundamental skill for everyone,
not just for computer scientists”.

adapted from Wing (2006)

Wing (2006) underlined that computational thinking (CT) is not only a programming skill employed by computer scientists but also a daily living skill required by everyone. In reality, we employ CT unknowingly in our daily lives. If you lose something, for instance, you will recall the procedures to search for it. CT was first introduced by Papert (1988), and its definition, teaching, and evaluation have been debated by several scholars ever since (Grover & Pea, 2013). Wing (2008) argued that CT integrates mathematics and engineering by applying the design of systems that help humans solve complex problems (Lu & Fletcher, 2009). Current educators must establish and promote facilities for learning computer technology.

CT is a collection of teaching practices that have been shown to benefit STEM students, regardless of their technological interests or backgrounds (Grover & Pea, 2013; Honey et al., 2014). To accomplish this goal, it has been proposed to integrate computer principles into core curricular areas by using "computational thinking language." The CT approach is useful for students to develop solid mental models and allows for a larger range of subjects to be addressed (Barr & Stephenson, 2011). CT is generally used to improve students' problem-solving skills and talents when they begin to think creatively. Lu and Fletcher (2009) proposed that students who had learn CT early and often, with a focus on these two concepts: "uses computational processes to produce virtual artefacts, not their manifestations in specific coding languages" and "skills for abstracting and expressing information." Meanwhile, gamification transforms students into active participants in the use of digital technology (Resnick et al., 2009). Abstracting and modularization, testing and debugging, experimenting and iterating, reusing and combining are all computational practices used in computer programming. Aho (2012) further argued that CT entails problem framing so that solutions can be stated as computer steps and algorithms.

CT language is defined as "vocabularies and symbols that can be used to annotate and describe computation, abstraction, and information, as well as providing notation for semantic interpretation of computational processes" (Barrows & Tamblyn, 1980; Lu & Fletcher, 2009). CT's main characteristics are around abstraction, automation, and analysis, which is critical for understanding how learners might utilize CT for problem formulation. The definition of abstraction is "identifying and removing significant information to define fundamental ideas." In problem solving, abstraction is described as reducing a problem to its essence. While abstraction refers to the process of collecting generalized features or activities into a single set that can be used to represent all specific cases while simplification refers to the process of reducing simplified characteristics or actions into a single set that can be used to represent all specific cases (Wing, 2008). Automation is a labor-saving procedure in which a computer or machine performs repetitive tasks that require more processing capacity than a person. Analysis is a reflective technique including the implementation and evaluation of a solution. In a nutshell, CT entails "solving problems, building systems, and comprehending human behavior using core computer science concepts" (Wing, 2006).

Literature Review: Problem Solving

Polya (1973) who pioneered in the problem solving, quoted that "solving a problem is similar as finding a way out of a difficulty, a way around an obstacle or attaining an aim that is not immediately understandable." Meanwhile, Green, Alison and Gilhooly (2005) defined "problem solving, in all of its forms, is an activity that meaningfully structures daily life." As a result, the type of problem dictates the type of cognitive skill required to solve the problem. For example, language skills are used to read about a problem and debate it, whereas memory skills are used to recall prior knowledge, and so on.

In general, the facts discovered that teachers do not adequately account for the fact that word problem solving proficiency has many intertwined and interdependent skills, including problem-solving, reasoning, modelling, handling symbols and formalism, representation, communication, utilizing of resources and softwares (Jaworski, 2015). Despite these significant inter-relationship characteristics, many mathematic teachers prefer to focus on one aspect at a time, incorrectly expecting other aspects to develop as a result of their intention (Kilpatrick & Swafford, 2002). When teachers have good problems to teach their students (for example concepts and relationships between mathematical ideas), but they delivered them poorly or inadequately (e.g. procedures and basic computational skills) (Stigler & Hiebert, 2004). Pedagogical factors also affect children's understanding towards word problem (Lean et al., 1990). Students' perspectives on what constitutes a word problem indicate that learning and practicing differ greatly from ideals (Jimenez & Verschaffel, 2014). As such, there is a need for instructional strategy to train students to think critically and to assist them in the development of word problem solving skills with new mathematical skills and concepts (Verschaffel et al., 2000).

As a result, mathematics education faces a significant challenge in establishing and connecting various aspects, skills, activities, and motivations in word problem solving. It is believed that the development and integration of conceptual and procedural mathematical knowledge and skills is a critical component of this challenge. Moreover, word-problem solving constitutes one of the most important mediums through which students can potentially learn to select and apply strategies necessary for dealing with everyday problems. It is also worth bearing in mind that various interlinked motives and activities have contributed to the phylogenesis of mathematical knowledge, such as play, find, order, application, construction, argumentation, evaluation, and calculation.

According to this study, 21st-century mathematical literacy includes mathematical reasoning and computational thinking to aid in problem solving. It is also aware that students should have and be able to demonstrate computational thinking skills in the PISA 2021 program as part of their problem-solving practice, owing to the increasing and evolving role of computers and computer tools in both day-to-day and mathematical literacy problem-solving contexts. In most studies, computational thinking is related to problem solving, where they acknowledged computational thinking as parts of a problem-solving activity. For example, Wing (2006, 2008) incorporated solving problems using computer science concepts in her definition of computational thinking. Her computational thinking skills are including pattern recognition, designs and uses of abstraction, pattern decomposition, determining where uses computational tools to perform analysis or solving a problem, followed by recognizing algorithms as part of a comprehensive solution. The framework anticipates that by emphasizing the importance of computational thinking as it relates to mathematics, the participating countries will focus on the role of computational thinking in their curricula and pedagogies.

Computing is becoming an increasingly significant instrument for doing scientific research. Computational thinking is a type of analytic thinking that uses mathematical and engineering concepts to analyze and solve tough real-world issues. This term is first used by Papert (1996), who is well known for the development of the Logo software. When it comes to solving complex real-world technological problems, computational thinking, and its components, as described by Liu and Wang (2010), are essential modes of thought. To solve the word problems, they combined critical thinking with prior knowledge and then applied the resulting construct. Computational thinking does not require word problems to be solved in the same way that a computer does, but it does promote critical thinking by utilizing computer science principles and techniques. As a result, when students face advanced technological challenges in the real world, computational thinking is a prerequisite for problem solving.

Papert (1980) who pioneered that explore alternate methods of teaching difficult problem-solving abilities and delivering dynamic learning experiences have shown a great deal of interest in coding in the classroom. (Kalelioglu, 2015; Lye & Koh, 2014). (Voskoglou & Buckley, 2012) also stated that critical thinking and creativity in learning are necessary for knowledge acquisition and application to solve problems when confronted with complex real-world technological problems. As a results, Haapasalo and Zimmerman (2015) summarized the problem-solving, reasoning and proof, communication, connection and representation as the five well known process standards for acquiring and using mathematics knowledge for students to engage and explore in problem solving strategies in mathematics curriculum.

Literature Review: Problem Based Learning

Because subject-based instruction may not be the most effective way to prepare students for future professional education, problem-based learning (PBL) was developed (Boud, 1985). Barrows and Tamblyn (1980) pioneered the problem-learning learning (PBL) technique in the 1960s. PBL is part of a long tradition of experiential learning centered on the investigation, justification, and resolution of significant problems. (Barrows, 2000; Torp & SageSara, 1998). Students learned in PBL by solving challenges and reflecting on their experiences (Barrows & Tamblyn, 1980). Because it emphasizes learning in the context of real-world circumstances and challenges students to take responsibility for their own education, PBL is great for assisting students in becoming active learners.

PBL motivates students to integrate information from core disciplines to deal with real-world situations by using ill-structured challenges as a learning starting point (Hmelo-Silver, 2004; Wilkerson & Gijselaers, 1996). PBL is defined as an instructional technique in which students gain information through guided problem-solving on a complicated issue with no final solution. PBL instruction encouraged the integration of existing knowledge as well as the skills required for lifelong learning. PBL is defined as an instructional technique in which students gain information through guided problem-solving on a complicated issue with no final solution. PBL instruction promoted the integration of prior knowledge and the abilities required for lifelong learning.

Learners acquire wide and adaptable knowledge and cross-disciplinary learning facts in order to develop critical competence. A domain's knowledge attainment is systematically arranged around its fundamental principles (Chi et al., 1981). It emphasizes both helping students create techniques and constructing knowledge (Collins, Brown, & Newman, 1987; Hmelo & Ferrari, 1997; Kolodner et al., 2003). Few scholars have reached a consensus that PBL is an effective instructional strategy for fostering transferable problem-solving and deductive reasoning skills in pupils. (e.g., (Albanese & Mitchell, 1993; Dabbagh & Denisar, 2005; Strobel & van Barneveld, 2009). Patel et al (1991, 1993) found that PBL learners are more likely than traditional learners to apply hypothesis-driven reasoning to understand clinical challenges. PBL students have been educated in hypothesis formation through the problem-solving process. Hmelo-Silver (2004) noted that Students in PBL identify what they need to know and acquire to solve an problem, apply prior knowledge and new topic knowledge to the problem, and then evaluate on what they've learned and the efficacy of the tactics used. Students successfully work in small groups in the majority of PBL implementations. Each individual must collaborate well with others in order to share information and contribute to the group's problem-solving decision-making (Schmidt &

Moust, 2000). Responsibility for the problem's solution as well as the learning process increases the intrinsic motivation of students. This is a major advantage of PBL (Savery & Duffy, 1996). The summarized Barrows's problem-based learning cycle is represented in Figure 1.

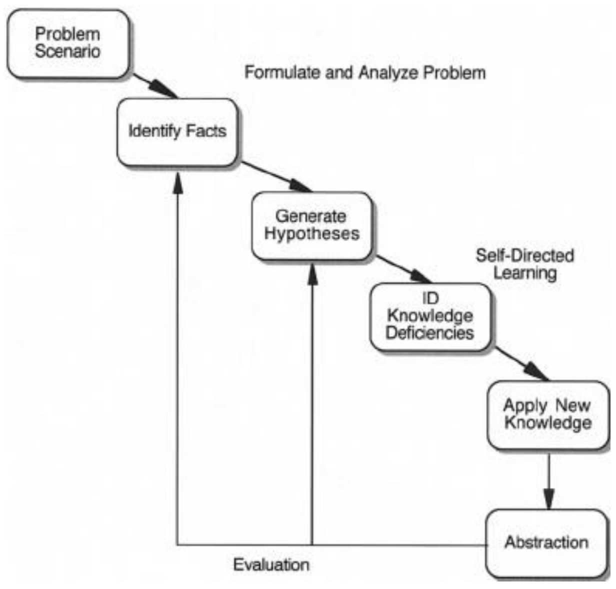


Figure 1. The problem-based learning cycle *(Barrows, 2000)*

PBL's fundamental promise is the enhancement of problem-solving abilities. The majority of PBL research findings support this hypothesis. Gallagher et al (1992) conducted an experiment using an interdisciplinary PBL course titled Science, Society, and the Future (SSF) with a comparative group of gifted high school students. From pre-test to post-test, they discovered that PBL students significantly increased their usage of problem-finding, a crucial problem-solving method. In contrast, during the post-test, the comparison group tended to bypass the problem-finding step and proceed directly to the implementation step. Results indicated that PBL is successful at fostering the development of suitable problem-solving processes and skills among students. In addition, PBL has a positive effect on students' capacity to apply fundamental science knowledge and translate problem-solving skills to professional or personal settings. Woods (1996) reported that employers praised the strong problem-solving skills and employment performance of PBL chemical engineering graduates from McMaster University. In contrast to other new employees, who normally take one to one and a half years of on-the-job training to be able to independently solve problems, PBL graduates are capable of independent problem solving upon graduation.

Methodology: Design Pedagogy Bridging Computation and Mathematics

This method was created with the preferences and skills of students in today's schools in mind. Table 1 summarizes the theoretical framework of computational mathematics problem-based learning (CM-PBL), which defines problem solving competency as an individual's ability to effectively engage mathematical equations (mathematical models) in a computer relation (simulation models); whereby more problem-solving strategies attempt to solve a problem by a corresponding software model and gaining new knowledge required to arrive at a solution, as well as pooling their prior knowledge. Table 1: Computational mathematics problem-based learning theoretical framework (CM-PBL)

- CT proposed by Wing (2006) that engagement of problem breakdown, problem reformulation, recursion, abstraction, and systematic testing are five cognitive processes that are critical to solving problems creatively and successfully.
- Barrows (2000) who promoted the mathematics education, problem-based learning cycle was encouraged, problem-solving abilities were developed, and social collaboration was increased.
- Robertson and Howells (2008) proposed that, rather than when students create or redesign computer games, they learn more when they play educational computer games.
- Lappan et al (1986) designed A move from show-and-do to problem-based learning that is capable of fostering in-depth and interconnected mathematical comprehension in children.
- Wing (2006) proposed that teaching CT as a foundational skill on par with arithmetic involves emphasizing algorithmic thinking and the capacity to apply mathematical concepts to create more effective, equitable, and secure solutions.
- According to Resnick (2013) who encouraged the use of programming in addition to learning problem-solving, design, and communication skills in addition to mathematics and computational ideas. He emphasized that these abilities are available to everyone, regardless of 21st century capabilities.

Table 1
Computational Mathematics Problem Based Learning (CM-PBL) Theoretical Framework

	Abstraction	**Automation**	**Analysis**
Understand and Identify Problems	Ignore the details and focus on the important facts and properties by breaking down the problems into manageable chunks.	Determine the interaction between rules and a computer that underlies the problems.	-Is the correct abstraction made? - Are the predicted relationships between prior mathematical knowledge and the task accurate?
Devising a Plan	Generating ideas about potential solutions by modelling the main characteristics and patterns of problems using the concept and identifying objectives	Reframe a heuristic problem as one that can be solved utilizing digital and stimulation tools to automate problem solutions	Are there any strategies that have gone unnoticed?
Execute	Identify the problem's gaps in knowledge and seek out new information.	Construct a system based on the problem-solving and prior information.	Do the mathematical principles incorporated into the system allow for the solution of problems?
Review and Reflection	Abstracting and fixing new information acquired through solution development.	Take purposeful activity to derive solutions by employing a computer	Evaluate and provide feedback on the effect that knowledge has on the process of problem resolution.

As a result, it is proposed that mathematics education in higher education be taught in a digitally improved learning environment. CM-PBL module intended to teach students how to solve problems and construct their own videogames using mathematical concepts. Problem-solving strategies, mathematical ideas, coding concepts, and the software development process were all highlighted in CM-PBL. According to research on the use of technology-enhanced and problem-based learning approaches to help students' mathematical knowledge and application, CM-PBL is used to foster problem-solving skills. CM-PBL enables students to use computers to improve their problem-solving skills by combining mathematical principles into a series of puzzles, allowing for the development of conceptual understanding. By combining the theories of computational thinking, mathematical thinking, and problem-based learning, this research presents CM-PBL pedagogy.

Elements of The Proposed Pedagogy

When CM-PBL is utilised across several curriculum areas, computational thinking (CT) influences how students' approach and solve problems. CT makes problem-solving easier by offering various approaches to a given problem (Sneider et al., 2014). Students' confidence in their academic abilities will be strengthened if they are given the coding skills necessary to design novel or novel solutions to challenges. Learners are exposed to different techniques to design their own learning process by viewing content units from a critical vantage point, which is a potent learning experience that promotes information and knowledge transfer (Strampel & Oliver, 2007). CM-PBL gives students more possibilities to learn to code and apply their math skills to the production of their own projects using a variety of coding tools. Coding is regarded as a process that includes problem-solving techniques (Gomes & Mendes, 2007). Meanwhile, learning coding steps is able to develop problem solving skills (Antonakos, 2011). Knowing and implementing solely programming codes is insufficient; CT is required only when a specific issue is detected. The use of CM-PBL during problem-solving procedures has an impact on students' problem-solving abilities.

According to Lee et al (2011) as shown in Figure 2, during the "Use" phase of this development, students interact with an existing computational artefact. The "Modify" stage teaches students how to gain computational thinking abilities (such as writing algorithms as computer instructions) by customizing and improving on the ideas of others. Students may be encouraged to design new computational projects that address problems of their choosing as they gain proficiency and confidence (the "Create" stage). The three pillars of computational thinking, abstraction, automation, and analysis, all come into play throughout the "create" process. It is critical to maintain a level of challenge that fosters growth while decreasing anxiety throughout this transition.

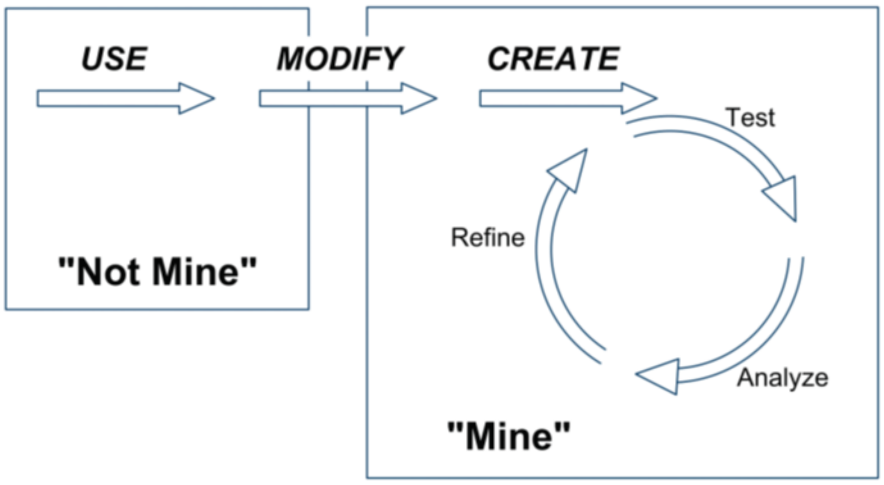


Figure 2. Use–Modify–Create Learning Progression *(Lee et al., 2011)*

As students' skills and, consequently, capacities grow, they face increasingly difficult design challenges. Activities that were previously "too challenging" and anxiety-inducing become tolerable by introducing appropriate, gradually harder experiences. If a student has constructed a UML model, he will find it easier to "translate" that model into visual programming. As a result, if a student understands the model's content and technique, it is simple to learn the syntax of any language. Visual programming with flowcharts is the best way for beginners to construct general-purpose software (Charntaweekhun & Wangsiripitak, 2006).

The following example is about how to solve a word problem: Carson is 7 years old, and Isaac is 9 years old. How old will Isaac be when Carson is 14 years old? Here we see the steps of solving the task: UML activity diagram (Figure 3), and the script from Scratch project implementing the coding to find the solution (Figure 4).

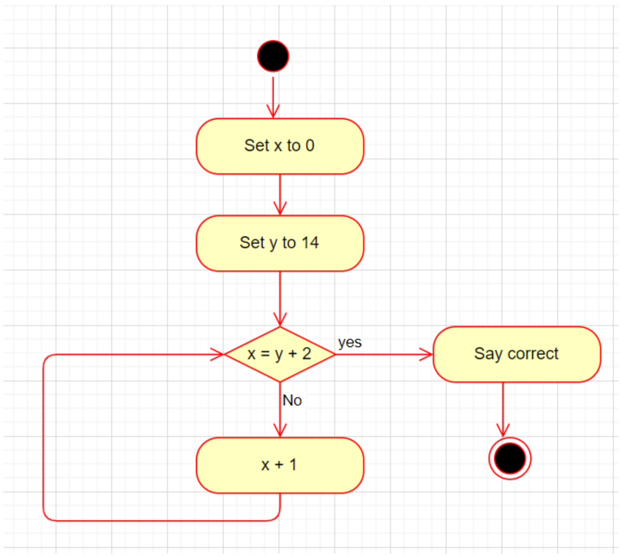
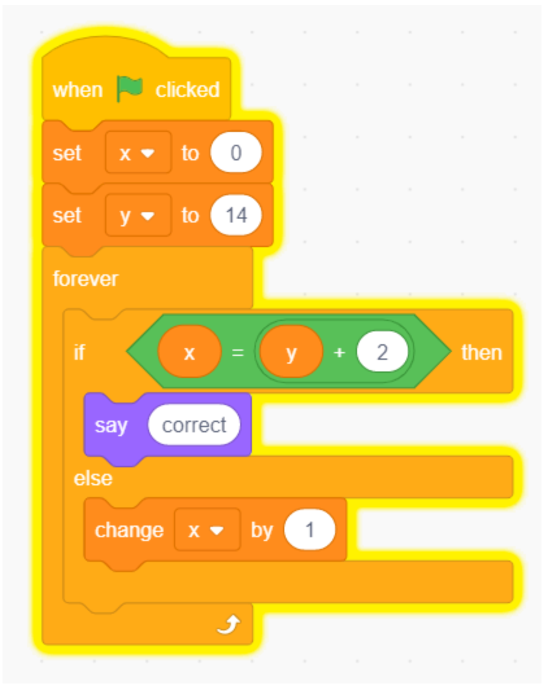


Figure 3. UML Activity diagram



```
when [flag] clicked
set x to 0
set y to 14
forever
    if x = y + 2 then
        say correct
    else
        change x by 1
```

Figure 4. Scratch Project

It is believed that CM-PBL pedagogy is vital for the development of problem-solving abilities and other 21st century talents such as critical thinking, creativity, computational thinking, social-intercultural skills, communication and teamwork, productivity, leadership, and responsibility (Durak & Guyer, 2018; Lau & Yuen, 2011). CT is a crucial collection of knowledge, skills, and attitudes for problem-solving and an important skill in the process of learning to code, according to this study. CT requires understanding and describing the problem, locating suitable solutions, and cognitive patterns such as reflecting and abstracting (Wing, 2006). This technique is effective in aiding the development of various high-level cognitive skills and knowledge areas, hence facilitating students' computer skills acquisition. In order to design processes and solve problems using computers, CT requires extra abilities such as data analysis, modelling, and data summarization.

Discussion and Conclusions

Based on STEM, mathematics integration, and computer programming, this study presented a novel way to mathematics instruction. It is thought that creativity and enjoyment are the keys to affective motivation, which results in a shift in attitude toward mathematics. An integrated computer system and dynamic mathematics environment can be improved by building game implementations that give a constructionist setting for learning mathematical topics. The goal of CT, on the other hand, is to integrate computer coding and problem-solving skills into all disciplines. The significance of technology in education requires immediate consideration. The CT curriculum must be implemented beginning in elementary school and continuing through higher education. CT must become one of the four primary talents writing, reading, arithmetic, and CT, in order to be properly integrated. This study will be beneficial to educators who want to integrate word problem solving in mathematics with computing and improve CT and problem-solving skills.

Acknowledgement

The authors would like to express their gratitude for the financial aids and supports provided by Universiti Putra Malaysia (UPM) through the research grant number GP-IPS/2020/9683300.

Corresponding Author

Ku Soh Ting
Department of Science and Technical Education, Faculty of Educational Studies, 43400 Universiti Putra Malaysia, Serdang, Selangor
Email: ting_blue2003@yahoo.com

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 833–835. https://doi.org/10.1093/comjnl/bxs074

Albanese, M. A., & Mitchell, S. (1993). Problem-based learning: A review of literature on its outcomes and implementation issues. *Academic Medicine*, *68*(1), 52–81. https://doi.org/10.1097/00001888-199301000-00012

Allan, V., Barr, V., Brylow, D., & Hambrusch, S. (2010). Computational thinking in high school courses. *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 390–391. https://doi.org/10.1145/1734263.1734395

Antonakos, J. L. (2011). *Computer Technology and Computer Programming: Research and Strategies*. Apple Academic Press, Inc.

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

Barrows, H. S. (2000). *Problem-based Learning Applied to Medical Education* (Revised Ed). Southern Illinois University School of Medicine. https://books.google.com.my/books?id=L6ZXAAAACAAJ&dq=Problem-Based+Learning+Applied+to+Medical+Education&hl=en&sa=X&ved=0ahUKEwiC7IbF0fnjAhVk73MBHQ1PCtYQ6AEILjAB

Barrows, H. S., & Tamblyn, R. M. (1980). *Problem-Based Learning: An Approach to Medical Education*. Springer Publishing Company, Inc.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Boud, D. (1985). Problem-based learning in perspective. In D. Boud (Ed.), *Problem-based learning in education for the professions* (pp. 13–18). Higher Education Research and Development Society of Australasia.

Charntaweekhun, K., & Wangsiripitak, S. (2006). Visual Programming Using Flowchart. *International Symposium on Communications and Information Technologies*, 1062–1065.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and Representation. *Cognitive Science*, *5*(2), 121–152. https://doi.org/10.1207/s15516709cog0502_2

Collins, A., Brown, J. S., & Newman, S. E. (1987). Cognitive Apprenticeship: Teaching the craft of reading, writing, and mathematics. In *Center for the Study of Reading*.

Dabbagh, N., & Denisar, K. (2005). Assessing team-based instructional design problem solutions of hierarchical versus heterarchical web-based hypermedia cases. *Educational Technology Research and Development*, *53*(2), 5–22.

Durak, H. Y., & Guyer, T. (2018). Design and development of an instructional program for teaching programming processes to gifted students using scratch. In J. Cannaday (Ed.), *Curriculum Development for Gifted Education Programs* (1st ed., pp. 61–99). IGI Global.

Felleisen, M., & Krishnamurthi, S. (2009). Viewpoint: Why computer science doesn't matter. *Communications of the ACM*, *52*(7), 37–40. https://doi.org/10.1145/1538788.1538803

Gallagher, S. A., Stepien, W. J., & Rosenthal, H. (1992). The Effects of Problem-Based Learning On Problem Solving. *Gifted Child Quarterly*, *36*(4), 195–200. https://doi.org/10.1177/001698629203600405

Gomes, A., & Mendes, A. J. (2007). Learning to program - difficulties and solutions. *ICEE 2007-International Conference on Engineering Education*. https://www.researchgate.net/publication/228328491_Learning_to_program_-_difficulties_and_solutions

Green, Alison, J. K., & Gilhooly, K. (2005). Problem solving. In N. Braisby & A. Gellatly (Eds.), *Cognitive Physcology* (1st ed., pp. 347–381). Oxford University Press.

Grover, S., & Pea, R. D. (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

Haapasalo, L., & Zimmerman, B. (2015). Investigating mathematical beliefs by using a framework from the history of mathematics. In C. Bernack-Schler, R. Erens, T. Leuders, & A. Eichler (Eds.), *Views and Beliefs in Mathematics Education* (pp. 197–211). Springer.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, *41*(1), 183–187. https://doi.org/10.1145/1539024.1508931

Hmelo-Silver, C. E. (2004). Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review*, *16*(3), 235–266. https://doi.org/10.1023/B:EDPR.0000034022.16470.f3

Hmelo-Silver, C. E., & Ferrari, M. (1997). The problem-based learning tutorial: Cultivating higher order thinking skills. *Journal for the Education of the Gifted*, *20*(4), 401–422. https://doi.org/10.1177/016235329702000405

Hoffer, B. M. (2012). Satisfying STEM Education Using the Arduino Microprocessor in C Programming. *ProQuest Dissertations and Theses*, 220.

http://login.proxy.library.vanderbilt.edu/login?url=http://search.proquest.com/docview/1069255002?accountid=14816%5Cnhttp://sfx.library.vanderbilt.edu/vu?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&genre=dissertations+%26+theses&sid=

Honey, M., Pearson, G., & Schweingruber, H. (2014). *STEM integration in K-12 education: Status, prospects, and an agenda for research*. National Academies Press.

Jaworski, B. (2015). Mathematics meaning-making and its relation to design of teaching. *PNA*, *9*(4), 261–272.

Jimenez, L., & Verschaffel, L. (2014). Development of children's solutions of non-standard arithmetic word problem solving non-standard arithmetic word problems. *Revista de Psicodidáctica*, *19*, 93–123.

Kalelioglu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200–210. https://doi.org/10.1016/j.chb.2015.05.047

Kilpatrick, J., & Swafford, J. (2002). *Helping children learn mathematics*. National Academies Press.

Kolodner, J. L., Camp, P. J., Crismond, D., Fasse, B., Gray, J., Holbrook, J., Puntambekar, S., & Ryan, M. (2003). Problem-Based Learning Meets Case-Based Reasoning in the Middle-School Science Classroom : Putting Learning by Design (tm) Into Practice. *Journal of the Learning Sciences*, *12*(4), 495–547. https://doi.org/10.1207/S15327809JLS1204_2

Langdon, D., Mckittrick, G., Beede, D., Khan, B., & Doms, M. (2011). *STEM: Good Jobs Now and for the Future*. https://www.purdue.edu/hhs/hdfs/fii/wp-content/uploads/2015/07/s_iafis04c01.pdf

Lappan, G., Philips, E., Winter, M. J., Shroyer, J., & Fitzgerald, W. (1986). *Middle grades mathematics project: Probability*. Addison-Wesley Publishing Company.

Lau, W. W. F., & Yuen, A. H. K. (2011). Modelling programming performance: Beyond the influence of learner characteristics. *Computers and Education*, *57*(1), 1202–1213. https://doi.org/10.1016/j.compedu.2011.01.002

Lean, G. A., Clemens, M. A., & Del Campo, G. (1990). Linguistic and pedagogical factors affecting children's understanding of arithmetic word problems: A comparative study. *Educational Studies in Mathematics*, *21*, 165–191.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

Liu, J. L., & Wang, L. H. (2010). Computational Thinking in Discrete Mathematics. *2010 Second International Workshop on Education Technology and Computer Science*, 413–416.

Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. *SIGCSE Bulletin Inroads*, *41*(1), 260–264. https://doi.org/10.1145/1539024.1508959

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

Ministry of Education Malaysia. (2013). *Malaysia Education Blueprint 2013-2025: Preschool to Post-Secondary Education*. https://www.moe.gov.my/muat-turun/penerbitan-dan-jurnal/dasar/1207-malaysia-education-blueprint-2013-2025/file

Papert, S. (1980). *Mindstorms: Children, Compuers, and Powerful Ideas*. Basic Books Inc.

Papert, S. (1988). A critique of technocentrism in thinking about the school of the future. *Children in the Information Age*, 3–18.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95–123.

Patel, V. L., Groen, G. J., & Norman, G. R. (1991). Effects of conventional and problem-based medical curricula on problem solving. *Academic Medicine*, *66*(7), 380–389.

Patel, V. L., Groen, G. J., & Norman, G. R. (1993). Reasoning and Instruction in Medical Curricula. *Cognition and Instruction*, *10*(4), 335–378. https://doi.org/10.1207/s1532690xci1004_2

Polya, G. (1973). *How To Solve It Mathematical Method*. Princeton University Press.

Resnick, M. (2013). *Learn to Code , Code to Learn*. EdSurge. https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn

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

Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers and Education*, *50*(2), 559–578. https://doi.org/10.1016/j.compedu.2007.09.020

Savery, J. R., & Duffy, T. M. (1996). Problem based learning: An instructional model and its constructivist framework. In B. Wilson (Ed.), *Constructivist Learning Environments: Case Studies in Instructional Design* (pp. 135–148). Educational Technology Publications.

Schmidt, H. G., & Moust, J. H. C. (2000). Factors affecting small-group tutorial learning: A review of research. In D. H. Evenson & C. E. Hmelo (Eds.), *Problem-based learning: A research perspective on learning interactions* (pp. 19–51). Lawrence Erlbaum Associates Publishers.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational Thinking in High School Science Classrooms: Exploring the Science “Framework” and “NGSS.” *Science Teacher*, *81(5)*, 10–15. https://doi.org/10.2505/4/tst14

Stigler, J., & Hiebert, J. (2004). Improving mathematics teaching. *Educational Leadership*, *61*(5), 12–17.

Strampel, K., & Oliver, R. (2007). Using technology to foster reflection in higher education. *Proceedings Ascilite Singapore 2007 - ICT:Providing Choices for Learners and Learning*, 973–982. http://www.ascilite.org.au/conferences/singapore07/procs/strampel.pdf%0A

Strobel, J., & van Barneveld, A. (2009). When is PBL More Effective? A Meta-synthesis of Meta-analyses Comparing PBL to Conventional Classrooms. *Interdisciplinary Journal of Problem-Based Learning*, *3*(1), 44–58. https://doi.org/10.7771/1541-5015.1046

Torp, L., & Sage, S. (1998). *Problems as Possibilities: Problem-Based Learning for K-12 Education* (2nd Ed.). Association for Supervision and Curriculum Development.

Verschaffel, L., Greer, B., & de Corte, E. (2000). *Making sense of word problems*. Swetz and Zeitlinger Publishers.

Voskoglou, M. G., & Buckley, S. (2012). Problem Solving and Computational Thinking in a Learning Environment. *Egyptian Computer Science Journal*, *36*(4), 28–46. http://arxiv.org/ftp/arxiv/papers/1212/1212.0750.pdf

Wilkerson, L., & Gijselaers, W. H. (1996). Concluding Comments. *New Directions for Teaching and Learning*, *68*, 101–104. https://doi.org/10.1002/tl.37219966814

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

Woods, D. R. (1996). Problem-based learning for large classes in chemical engineering. *New Directions for Teaching and Learning*, *1996*(68), 91–99. https://doi.org/10.1002/tl.37219966813