STUDENTS ENGAGING IN MATHEMATICAL PROBLEM-SOLVING THROUGH COMPUTATIONAL THINKING AND-PROGRAMMING ACTIVITIES: A SYNTHESIS OF TWO OPPOSITE EXPERIENCES

Said Hadjerrouit and Nils Kristian Hansen
University of Agder, Institute of Mathematical Sciences, Kristiansand, Norway

ABSTRACT

This paper aims at exploring students' experiences when engaging in mathematical problem-solving through computational thinking and programming by a combination of theoretically derived insights and task-based activities. The main method used is a semi-structured interview with two undergraduate students who were presented with a mathematical task to solve while responding to questions about the mathematical solving process. The results reveal two students' opposite experiences. While one struggled to get acquainted with the task and the associated programming activity, the other was able to handle and solve the task using computational thinking (CT) and programming skills very easily. Conclusions are drawn from the students' experiences to promote mathematical problem-solving through computational thinking and programming in mathematics education at the undergraduate level.

KEYWORDS

Algorithm, Computational Thinking (CT), Matlab, Mathematical Problem-Solving, Programming, Usability

1. INTRODUCTION

In parallel to emerging programming languages (Wing, 2014), students in mathematics study programmes are expected to acquire basic algorithmic and computational thinking skills. These are considered important competencies for future work in society and should be acquired by all university mathematics students who could improve their mathematical problem-solving skills by benefitting from computational thinking and the power of programming languages (Shute, Sun, & Asbell-Clarke, 2017). It is assumed that with easier access to digital technology in universities, the integration of programming activities and computational thinking into teaching could be easier than in school mathematics.

This study explores students' experiences when engaging in mathematical problem-solving through CT and the programming language MatLab. The article is structured as follows. Firstly, the theoretical framing is outlined. Secondly, the context of the study, research question, methods, and the mathematical task given to the students are described. Then, the results are reported and analysed. This is followed by a discussion of the results. Finally, some remarks and future work conclude the article.

2. THEORETICAL FRAMING

Programming has had a relatively long history in mathematics education. Today, CT by means programming languages is becoming an important learning goal of mathematics education from primary to university level. It is therefore important to reflect on the conditions of integrating CT and programming skills into mathematics education and the importance of these skills in the work of today's mathematician (Broley, Caron, & Saint-Aubin, 2018). Given this introductory statement, the theoretical framing is outlined in three sections as follows.

ISBN: 978-989-8704-22-1 © 2020

2.1 Computational Thinking (CT): Literature Review

Several definitions of the term "CT" exist in the research literature. Wing (2014) describes CT as "the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out". Ejsing-Duun (2015) describes CT in similar words. It is the ability to work with algorithms understood as systematic and structured descriptions of problem-solving and construction strategies. Filho and Mercat (2018) define algorithmic thinking as the process of solving a problem step-by-step in an effective, non-ambiguous and organised way that can be translated into instructions to solve problems of the same type by an individual or a computer.

Wing (2008, 2014) points out that the main commonality between CT and mathematical thinking is problem-solving and structured construction process step-by-step. CT is also quite similar with engineering thinking in terms of design and evaluation of processes (Pérez-Marína et al., 2018). Moreover, CT and programming constructs such as variables and flow statements (if-then-else, for, and repeat) are closely connected to arithmetical and mathematical thinking (Lie, Hauge, & Meaney, 2017). This close connection between mathematics and CT might provide opportunities for mathematical problem-solving. In contrast, programming skills alone without the mediation of CT are important but may not be sufficient to improve mathematical problem solving. Thus, CT skills are critical for building efficient algorithms for mathematical problem-solving rather than trial-and-error and getting the program to run (Topallia & Cagiltay, 2018). In other words, CT requires students to be engaged in a continuously changing problem-solving process by designing effective algorithms that can be translated into efficient computer programs.

2.2 Mathematical Problem-Solving through CT and Programming

Drawing on the research literature (Kotsopoulos, et al., 2019; Lee, & Malyn-Smith, 2019; Romero, Lepage, & Lille, 2017; Santos et al., 2020; Weintrop, et al. 2016), the paper proposes a three-step approach to connect mathematical problem-solving to CT by means of programming languages. *Firstly*, students should have a good mathematical background to benefit from CT and programming languages. More specifically, they should be able to benefit from their knowledge to make sense of the task and have a mathematical understanding of it before formulating an algorithm and starting programming. *Secondly*, CT, in turn, should enable students to analyse and decompose the mathematical task and design an algorithm and how to perform it step-by-step before programming it. Engaging students in mathematical problem-solving through CT may enable a better understanding of mathematics beyond textbook mathematics and paper-pencil techniques. *Thirdly*, students should be able to translate the mathematical solution with the associated algorithm to the constructs of the programming language. This presupposes that the language is usable. Performing programming activities in mathematics education should provide opportunities to help students do mathematics and gain new knowledge that is otherwise difficult to acquire without experimenting with the program and thinking algorithmically. However, this might be difficult to achieve unless the mathematical tasks are well-designed, mathematically sound, correct, and faithful to the underlying mathematical properties.

2.3 Programming and Usability Issues

When referring to the notions of usability or usable technologies in mathematics education, the research literature focuses on educational software such as GeoGebra, CAS, and SimReal (Artigue et al., 2009; Bokhove & Drijvers, 2010; Hadjerrouit, 2019). However, programming languages differ from educational software and how they are used to implement mathematical problems where one can graph a function or compute an integral simply by entering the function and pressing a button in GeoGebra, for example. Hence, evaluating the usability of programming languages might not be as straightforward as it may seem. Still, three usability criteria can be applied to programming languages with slight modifications. Firstly, the extent to which the user interface of the programming language is easy to use and understand. Secondly, a usable programming language should allow a quick familiarisation with it in terms of learning the language constructs, such as variables, if-then-else, for and while loop, or repeat. The third criterion is the quality of feedback provided by the program in terms of semantic and syntax error messages, and whether these are useful to foster a successful implementation of the algorithmic solution through testing, correcting and improving the program.

A purely technical approach to programming will however not succeed unless students' engagement with mathematical problem-solving through CT is placed in a pedagogical context. A pedagogy-based approach to programming should enable a good degree of autonomy so that the students can work on their own and have a sense of control over their mathematical learning. Clearly, students should be able to acquire knowledge without being completely dependent on the teacher. Moreover, CT and programming languages should be a motivational factor for learning mathematics. In other words, CT should support students' engagement with problem-solving by means of intrinsically motivating tasks that are tied to the students' mathematical activities. Finally, interacting with a programming language when engaging in mathematical problem-solving should be supported by computational thinking skills through a structured construction of effective algorithms.

3. THE STUDY

3.1 Context of the Study and Research Question

This work is a single case study conducted in the context of a first-year undergraduate course on programming with applications in mathematics. The participants were two students from one class of eight students enrolled in the course in 2019. The students had varied knowledge background in mathematics, but no experience with the programming language being used in this course, that is MatLab. The course introduced the basic constructs of MatLab, that is variables, flow statements (loops, if-then-else, for, and repeat). These were then used to solve mathematical problems. The research question addressed in this paper is: *How do students engage in mathematical problem-solving through computational thinking and programming activities?*

3.2 Mathematical Task

The mathematical task presented to the students in this research is: The length of a curve may be approximated using Pythagoras' theorem by positioning a triangle adjacent to the curve (Figure 1, left, below). The length of the green line between A and B may then be approximated as $\sqrt{x^2 + y^2}$. The task is to write a MatLab function approximating the curve length of $f(x) = 2^x$ between two given x-values (Figure 1, right, below).

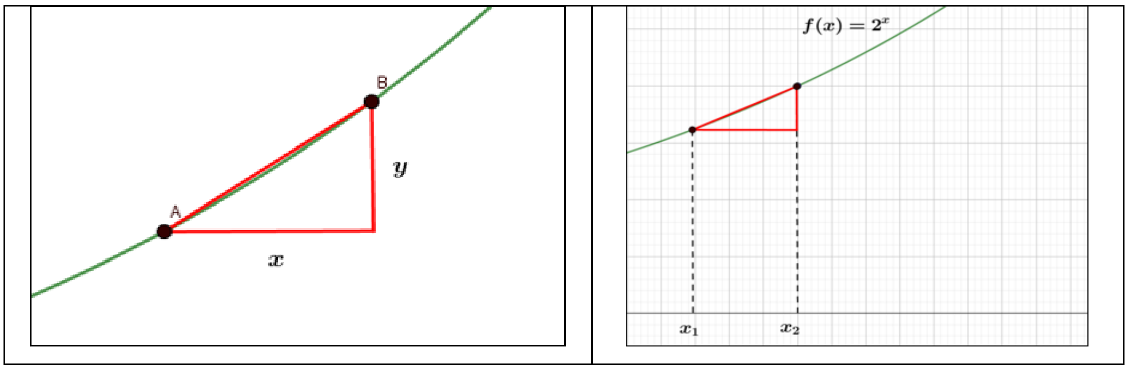

Figure 1. The mathematical task

The students were presented with the following skeleton of a MatLab function: function length=length estimate (x1, x2), length=? The students were then asked to enter the formula, based on x1 and x2, and replace the question mark. The MatLab function sqrt(x) can be used to calculate the square root √x.

3.3 Data Collection and Analysis Method

The main data collection method used is a task-based semi-structured interview of two students with different knowledge background in mathematics. The students were presented with a mathematical task to solve, while responding to questions in a dialogue with the teacher on the mathematical solving process, mathematical understanding, the role of CT and algorithms, possibilities and limitations of MatLab, feedback and interactions with MatLab, teacher help, and pedagogical considerations. Open-ended questions were also used to gain a deeper understanding of some important issues.

In terms of data analysis of the results, the three-step approach presented in the theoretical framing (section 2.2) served as a reference for analysing the students' problem-solving process, that is:

a) Understand the mathematical problem
b) Analyse and decompose the mathematical task, and then design an algorithm and how to perform it step-by-step before programming it
c) Finally, translate the algorithmic solution to the programming language code.

More specifically, the students were expected to solve the mathematical task presented to them in section 3.2. in three steps as follows:

a) Understand the task, that is, using Pythagoras' theorem to calculate the hypotenuse.
b) Formulate an algorithm, that is find the lengths of the triangle hypothenuse using the function $f(x) = 2^x$ and relate it to x_1 and x_2
c) Translate the algorithm into MatLab code, corresponding to: function length = lengthEstimate (x1, x2); length = sqrt((x2-x1)^2 + (2^x2 - 2^x1)^2)

The analysis of the results seeks indications of students' problem-solving through CT and MatLab according to this three-step approach. This is not the same as analysing and coding in the sense of grounded theory without theoretical background. Rather, it is an analytical tool that tries to address the research question about how students engage in mathematical problem-solving through computational thinking and programming activities drawing on students' interviews when solving the mathematical task. The interviews were transcribed and analysed according to an inductive strategy based on the interplay between the three-step approach to problem-solving and the empirical data collected using semi-structured interviews (Patton, 2002).

4. RESULTS

The results describe how the participating students engage in mathematical problem-solving through CT and MatLab. The students were given the task described in section 3.2, and paper and a pen to use. The abbreviations S1 and S2 are used for the students, and T for the teacher.

4.1 Student 1 (S1)

It took some time before the student made sense of the mathematical task. The teacher asked the student to develop a skeleton of the solution. The student did, and started thinking about the length, but suggested an incorrect solution, based only on the values of x1 and x2. After some calculation trials, the student noticed that the attempted solution was wrong. Then, the teacher encouraged the student to think computationally.

T: But before you start using MatLab, are you going to make an algorithm (…), for problem-solving before you start using MatLab?

S1: I just have to sit and think about it.

But still, the student continued guessing and calculating without thinking computationally. After an attempt to make sense of the task and calculate the length of the curve connecting it to the hypotenuse with a trial and error approach, the teacher provided a hint.

T: Now you say you know the hypotenuse and calculate y. But the hypotenuse is the unknown parameter here, the one you are supposed to calculate.

S1: Yes.

T: So now you have turned the problem around (…). It is just that that thought was a little backwards, maybe.

S1: Yes, it is quite possible.

Following this short dialogue, the student started using MatLab without developing an algorithmic solution or a clear strategy for solving the problem. The teacher then engaged in a discussion to guide the student step by step towards an algorithmic solution. Then, the teacher tested the function *“on zero and one and then I got 1.4*". Likewise, the student tested the formula and found 1.4142. The dialogue continued as follows:

T: (…). Do you have anything to say about (…) like that afterwards?

S1: No, I am, I was a little bit in doubt about how to (…) First, it was the task you asked about (…) and then it was (…) and then I thought (…) f(x) is the function in x2 would be that point minus the function of that point (…) that it would be the length. But that is where I was wrong, I felt (…) Because you meant it to be here (…) and I understand that now.

T: Yes, that is the point, (…). That is why you have to use Pythagoras to find (…). Did you think (…) There was a hint here, wasn't there? Square root?

S1: Yes, yes, yes, the square root (…). I knew it was probably wrong, but I just didn't quite understand what it was.

T: Well, because there was a clue there that you couldn't use, wasn't there. Then you realise that there is something (…)

This excerpt shows there is little indication that the student was following a clear problem solving strategy, which confirms that making sense of the problem and having a mathematical understanding of the task is of crucial importance before formulating an algorithm and starting programming, which is not the case in this excerpt. A few minutes later, the teacher asked the student if there is a tendency to favour pen and paper to solve the task algorithmically before starting using MatLab since developing an algorithm does not automatically require using the computer.

S1: If I have it in my head, sometimes I start with MatLab, and then I write some sort of sketch before going through it carefully. If I am not sure, I will start with paper.

T: Maybe the task was not quite clear?

S1: Yes, so far, but I had probably forgotten some of the principles there.

T: Principles related to MatLab or to the mathematical assignment?

S1: To the mathematical problem.

Again, the interview shows that the problem-solving process requires a good understanding of the mathematical task and computational thinking skills before programming it. Here again, the teacher reminded the student about the importance of algorithmic thinking before translating the task into MatLab code.

T: Now, the point of the assignment is that you should be able to translate the mathematical task into code in MatLab. That is really the point here.

S1: Yes, I felt that when I understood the mathematical thing, I had no trouble putting it into MatLab. It was simply that I had (…) forgotten a bit the length thing there. That f of that minus f of that is delta y, then.

4.2 Student 2 (S2)

After a short dialogue with the student on the menu structure of MatLab, which does not seem to be much appealing, the student started studying the assignment. The following excerpt shows making sense of the task to be solved is of crucial importance before formulating a solution by means of computational thinking. The teacher highlighted that these are the kind of tasks that have started to appear on the exams. So, it is very useful. After this short introduction, the student read the mathematical task described in section 3.2. Then, the teacher asked how to handle it.

T: Just, I have to ask you … How will you proceed to attack such a task? Will you go straight for the keyboard, and then …? You see, you have got a small skeleton here; will you go straight for the keyboard and start programming it, or will you …

S2: No, first I really try to think it through carefully, what the task is really about, here. And then I tend to make a draft as a first thing. (…) That I write down what the assignment actually asks for, and then that I make myself, in a way, a kind of design, then.

T: A design, yes.

S2: And what is it that the program is supposed to be able to calculate here, then? And then, when it is plainly clear to me, I would have gone into more depth on … on … what is somehow … how I am supposed to describe the code here. And here, in this case, there is a function, which is an estimate of length. And then I had to go in and have a look at the length. And then put it in a formula … Oh, I do not know if you want me to do the task itself, kind of.

T: (…) you do not have to spend so much (…) what we are a looking for? that is the thought process. You say you create a draft, do you write code then, or draw boxes, or?

S2: Sometimes I have drawn one of these … in a way a line, then. And then I have drawn a box. Or, if it involves if-statements, then I have drawn something a bit like this. (Shows sketch like figure 2, left, below.) If you look here. And then I have sort of a condition inside here, and then I have, if there is an *if* at the conditions, then I put this there (Shows sketch like figure 2, right, below, pointing to the new line.)

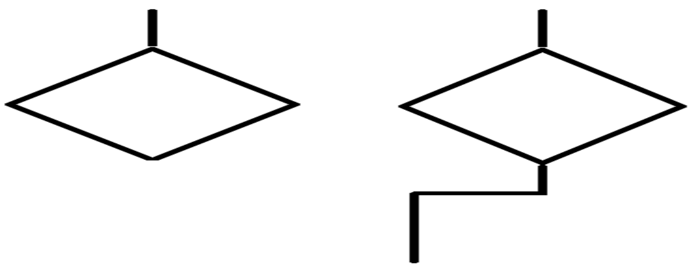
Figure 2. Student's sketches

T: That is what we call a flow chart.

S2: Flow chart, simply.

T: Yes, that is probably something you have learned once, maybe.

S2: Yes, I learned it in this study.

This excerpt shows that the student was aware of the importance of designing an algorithmic solution with a flow chart before starting programming the task, in stark contrast to student 1 who struggled to understand the mathematical task and formulating an algorithm. There are indications that the student understood how to proceed. Words like “design”, “draft”, “box”, and “assignment” point to algorithmic and computational thinking; and “if-conditions” and the associated sketch to programming code.

T: Yes, okay, yes. Very good, for it has turned out that people are very slow to adopt … They often go straight on to the programming. And that works fine when the task is easy, but when it gets a little more complex, then you need to split things up a bit.

S2: Yes, I had not thought about it until I, in a way, read it in this study, but after that I've started to … then I spend a bit of time in the process before I start typing the formula. If I have already started typing the formula, and then want to change it afterwards, then I have discovered that I do a lot of mistakes, then. So now I have tried to spend plenty of time on the preparatory work, so that I am one hundred per cent sure of what it is my function or program is supposed to do. And now I know it is a function I am supposed to write, and what is it … what is it I am actually supposed to calculate here?

Clearly, the second participant was not only able to spend time on some preparatory work, but the student was also reflecting on the function that is supposed to solve the task. Moreover, the student was aware of possible programming mistakes that could be made, and the time needed to correct them iteratively, and how the program will handle the errors and provide feedback to the student.

T: Yes, but that is really good. That is something we would love for more to be better at, so we should probably have stressed it more when teaching. You mention it yourself a bit, the algorithmic thinking. And that is a bit what we are looking for then, considering that algorithmic thinking is to become part of the subject of mathematics in school. So that is what we are trying to fish for, how are these students thinking?

S2: It is the kind of like you analyse this pretty carefully, and then you divide it, and then … in a way … I do not know, I work kind of structured, then, with one piece at a time.

T: Yes, but that is really good. It is a concept one has in programming, you make a design, then you divide it into pieces, where you can look at one piece at a time.

This way of computational thinking was confirmed by the teacher who pointed out to algorithmic thinking and structured design step-by-step, and decomposition of the task in small pieces, (…). Then, the student continued explaining the way of thinking algorithmically and computationally, including the programming process and the testing of the program.

S2: And then after I have made … made one, then … if it … now this task was a little bit different, then … but … but if I have made a program … after I have written the entire program code … If it typically is with for-loops and everything, then I always run through the program in my head. And write down that okay, now, *n* is equal to 1, What happens to that and that value. In that way I always get to put it to the test, and then I detect possible errors, then.

This last excerpt shows that the student is knowledgeable in some constructs of the programming language being used, the way the program code can be run and tested, and what happens if the program contains errors.

5. DISCUSSION AND CONCLUSIONS

The research question addressed in the paper is: *How do students engage in mathematical problem-solving through computational thinking and programming activities?* As one can see from the results, the participating students had two opposite experiences. They engaged differently in the mathematical problem-solving process. While the first student felt challenged by the mathematical task and the way to handle it, the second student made good use of CT, algorithms, and MatLab to solve the task.

Concerning student 1 (S1), the first challenge is the lack of mathematical knowledge, which hindered the student in making sense of the task, having a mathematical understanding of it, and then developing a problem-solving strategy that can be translated into an algorithm. According to the approach presented section 2.2, it seems that students should have a good mathematical understanding of the task as a basis for further development and use of computational and algorithmic thinking skills. The second challenge is related to the implementation of the algorithm using the constructs of the programming language. As the results show, student 1 struggled to use MatLab because of lack of background knowledge in the programming language. The student was therefore unable to relate it to the mathematical task using an algorithm. Clearly, the problem-solving process in three steps as described in the theoretical framing was challenging for the student.

Student 2 (S2) was more prepared to make use of CT than Student 1. Student 2 had sufficient mathematical knowledge and a good understanding of the task to make sense of it, and then design a structured problem-solving solution that can be translated into an algorithm. The excerpts show that the student had a good understanding of the task as a basis for further algorithmic development. Moreover, in contrast to student 1, student 2 had good background knowledge in the programming language MatLab and was able to relate it to the mathematical task using an algorithm. As a result, this student demonstrates that a combination of background knowledge in mathematics, algorithmic thinking, familiarity with the programming language are a pre-requisite to apply CT skills for mathematical problem solving.

These opposite students' experiences show that the minimum requirement for applying CT is a good combination of background knowledge in mathematics, algorithmic thinking skills, and familiarity with the programming language in question. These requirements should become integral parts of university mathematics education. To ensure success in integrating these elements, the pedagogical setting around mathematics courses should be well designed in terms of varied and intrinsically motivating tasks, student autonomy and differentiation. Moreover, as this study shows, the role of the teacher is still important to assist students in formulating solutions. Clearly, student autonomy cannot be fully expected for novices without good knowledge background in mathematics and familiarities with CT and programming. Hence, mathematical problem-solving should consider pedagogical modalities when using CT and programming practices.

Summarising, the study is limited to be generalised to all participating students. Hence, more elaborated and in-depth analysis with mixed methods and more participants is required to ensure more reliability and validity. Although the participating students are not representative for the average student enrolled in the course, two preliminary conclusions can be drawn from the study. Firstly, the connections between mathematics, CT, and programming languages are quite complex, and need to be clearly articulated and related in pedagogical settings. Secondly, engaging in mathematical problem-solving through CT and programming requires good background in mathematics, algorithmic thinking, and familiarity with the programming language constructs.

Future work will include all course participants and a mix of quantitative and qualitative methods to ensure more validity and reliability and assess the role and impact of CT in mathematics education in more depth.

REFERENCES

Artigue, M. et al. (2009). Connecting and integrating theoretical frames: The TELMA contribution. *International Journal of Computers for Mathematical Learning 14*, pp. 217-240.

Broley, L., Caron, F., & Saint-Aubin, Y. (2018). Levels of programming in mathematical research and university mathematics education. *Int. J. Res. Undergrad. Math. Ed. 4*, pp. 38–55.

Bokhove, K., & Drijvers, P. (2010). Digital tools for algebra education: Criteria and evaluation. *International Journal of Mathematics Learning 15*, pp. 45-62.

Filho, P., & Mercat, C. (2018). Teaching computational thinking in classroom environments: A case for unplugged scenario. *Conference: Re(s)sources 2018 - Understanding Teachers' Work Through Their Interactions with Resources for Teaching*. Lyon, France.

Hadjerrouit, S., & Gautestad, H.H. (2019). Evaluating the usefulness of the visualization tool SimReal+ for learning mathematics: A case study at the undergraduate level. In: D. Sampson et al. (Eds.), *Learning Technologies for Transforming Large-Scale Teaching, Learning, and Assessment*, (pp. 71-89). Springer Nature Switzerland AG.

Kotsopoulos, D., Floyd, L., Nelson, V., Makosz, S., & Senger, N. (2019). Mathematical or computational thinking? An early years perspective. In K. M. Robinson, H. P. Osana, and D. Kotsopoulos (Eds.). *Mathematical learning and cognition in infancy and early childhood: Integrating interdisciplinary research into practice*. New York: Springer.

Misfeldt, M., & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. In K. Krainer, & N. Vondrová (Eds.). *Proceedings of CERME9* (pp. 2524-2530). Prague, Czech Republic.

Lee, I., & Malyn-Smith, J. (2019). Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective. *Journal of Science Education and Technology.* https://doi.org/10.1007/s10956-019-09802-x

Lie, J., Hauge, I. O., & Meaney, T. J. (2017). Computer programming in the lower secondary classroom: Mathematics learning. *Italian Journal of Educational Technology*, *25*(2), pp. 27-35.

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2018). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior,* pp. 1-10.

Patton, M. Q. (2002). *Qualitative research & evaluation methods.* London: Sage Publications

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,* pp. 1-15.

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

Santos, S. C, Tedesco, P. A., Borba, M., & Brito, M. (2020). Innovative approaches in teaching programming: A systematic literature review. *Proceedings of the 12th International Conference on Computer Supported Education* (CSEDU 2020), Volume 1, pp. 205-2014. Prague, Czech Republic.

Shute, V.J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, pp. 1-10.

Weintrop, D. et al. (2016). Defining computational thinking for mathematics and science classrooms. *J Sci Educ Technol 25*, pp. 127–147.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*(1881), pp. 3717-3725.

Wing, J. M. (2014). Computational thinking benefits society. *Social Issues in Computing,* 40th Anniversary Blog, University of Toronto. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html