EXPLORING STUDENTS' COMPUTATIONAL THINKING FOR MATHEMATICAL PROBLEM-SOLVING: A CASE STUDY

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

ABSTRACT

The purpose of this paper is to investigate students' engagement in computational thinking (CT) and programming with MATLAB when solving a mathematical task in a programming course at the undergraduate level. The data collection method is participant observation of three groups of three students presented with a mathematical task to solve. The study uses a deductive-inductive analysis strategy based on the interplay between the theoretical framework and the empirical data. The results reveal that engaging students in CT for mathematical problem-solving is challenging for many reasons: Lack of mathematical thinking (MT), insufficient experience with CT, and more importantly lack of deeper connection between CT and MT. Conclusions are drawn from the results to promote mathematical problem-solving by means of CT and MT at the undergraduate level.

KEYWORDS

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

1. INTRODUCTION

Computational thinking (CT), mathematical thinking (MT) and programming are increasingly relevant aspects of undergraduate students in science studies. A primary motivation for introducing CT practices into mathematics classrooms is the rapidly changing nature of competencies of the discipline required for future work in society and the professional world. To benefit from CT, students need to investigate the way in which CT interacts with their reasoning about mathematical concepts. This study explores students' engagement in CT, MT and programming activities when solving mathematical problems.

The article is structured as follows. Firstly, the theoretical framework is outlined. Secondly, the context of the study, research question, methods, and task are described. Thirdly, the empirical results are analyzed and discussed. Finally, the limitations of the study and future work conclude the article.

2. THEORETICAL FRAMEWORK

Computational thinking (CT) represents a "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use'' (Wing 2006, p. 33). Accordingly, CT is a fundamental skill for virtually any discipline, including mathematics, physics, engineering, etc. It may have far reaching practical implications from which mathematics in particular can benefit.

2.1 Computational Thinking (CT)

Wing (2014) characterizes 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". Mistfelt & Ejsing-Duun (2015) describes CT as the ability to work with algorithms understood as systematic and structured descriptions of problem-solving and construction strategies. Likewise, Filho and Mercat (2018) defines algorithmic thinking as the process of solving a problem step-by-step in an effective, non-ambiguous and organized way that can be translated into instructions to solve problems of the same type by an individual or a computer. Csizmadia et al. (2015) emphasizes the role of logical reasoning in CT to make sense of problems through thinking clearly and precisely. It allows students to draw on their own knowledge to analyze problems, design algorithms, build, test, debug, and correct programs derived from the algorithm and associated solution to the problem. More precisely, CT means to engage in several cognitive processes with the goal of solving problems efficiently and creatively (Csizmadia et al., 2015; see also Wing (2006)). Firstly, it is the ability to think algorithmically, that is a way of getting to a solution to the problem step-by-step. Secondly, it is a way of thinking about problems in terms of decomposition of their components into manageable units that can be understood, solved, developed, and evaluated separately. Thirdly, CT is associated with generalization in terms of identifying patterns, similarities, and connections, and exploiting those features to generalize the problem-solving process to similar problems. In other words, generalization is a way of solving new problems based on previous solutions, building on prior experience, and generalizing these experiences. Fourthly, CT uses abstraction to make problems easier to think about. Abstraction is the process of making problems more understandable through reducing unnecessary details. Finally, CT makes use of evaluation, which is the process of ensuring that the problem-solving process, whether an algorithm or program, is fit for the purpose.

2.2 Computational Thinking, Mathematical Thinking, and Programming

Connecting CT and programming to MT in a meaningful way is at the heart of tackling mathematical problem-solving in a technological environment. The connection between CT and MT is not new and has a legacy of over 45 years in the theory of constructionism (Papert & Harel, 1991). Likewise, the connection of computer programming and CT with MT has been recognized since the development of the Logo programming language (Shodiev, 2015). In other words, CT and programming complement MT as a way of reasoning to solve mathematical problems (Wing, 2006). According to Shute, Sun, and Asbell-Clarke (2017, p. 145), MT consists of three parts: "beliefs about math, problem solving processes, and justification for solutions". MT involves the "application of math skills to solve math problems, such as equations and functions" (Ibid, p. 145). The authors conclude that MT and CT have a lot of communalities: problem solving, modelling, data analysis and interpretation, and statistics and probability (p. 145). As a result, with the use of CT and computers, the relationship between modelling, analysis, and solution of mathematical problems has changed, and it is becoming increasingly important (Buteau, et al., 2018).

The close connection between MT and CT provides opportunities for building efficient algorithms for mathematical problem-solving as a structured step-by-step construction processes that can be implemented using programming languages (Topallia & Cagiltay, 2018; Wing, 2008, 2014). More specifically, mathematical problem-solving through CT involves expressing a solution by means of decomposition in its parts, abstraction by removing unnecessary details, generalization from previous experiences, algorithmic thinking and transformation into a program that can be evaluated (Csizmadia et al, 2015). The challenge is to engage students in a mathematical problem-solving process through CT by designing effective algorithms that can be translated into efficient computer programs that can be tested, modified, and improved iteratively. There is also a clear connection between CT and coding, but CT is not the same as programming, but being able to program is a result of being able to think computationally (Shute, Sun, & Asbell-Clarke, 2017; Wing 2006). Rooted in these theoretical considerations, a model of mathematical problem-solving is elaborated inspired by Hadjerrouit and Hansen (2020). Figure 1 illustrates the model and demonstrates four points.

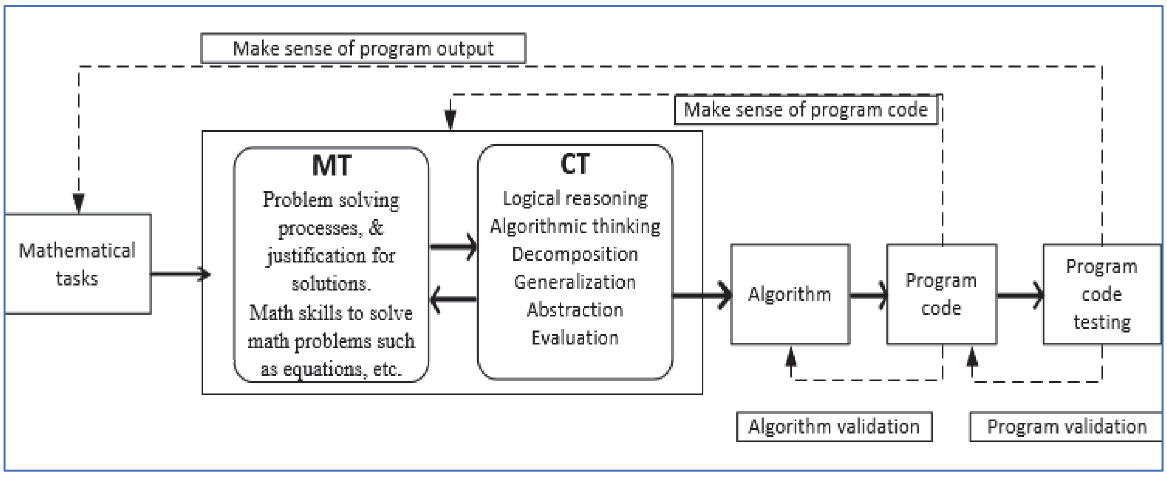

Figure 1. A model for mathematical problem-solving combining MT, CT, and programming and their interactions

Firstly, it is a pre-requisite that students have a good understanding of mathematical concepts and a capability for abstract reasoning and logical deduction to benefit from CT. Secondly, CT should in turn enable students to logically analyze, abstract, and decompose mathematical problems and design an algorithm step-by-step before programming it. Thirdly, students should be able to translate the mathematical solution and associated algorithm into programming code that can be tested and evaluated. Finally, the solution process should be generalized to a variety of similar problems.

This is not a linear model that starts from a mathematical problem and ends up with program code testing. It is rather an iterative model with feedbacks to previous processes to make sense of program output or validating the algorithm, etc. It is also particularly important to consider interactions between MT and CT. Using this model, the goal is to gather and analyze data on students' CT and MT, and programming activities. It is particularly important to look at the way students approach mathematical problem-solving with a range of MT skills in mind such as abstract reasoning and logical thinking, but without thinking computationally automatically, and how CT may help to decompose the problem and making it more understandable through abstraction, and generalization of the problem-solving process.

3. THE STUDY

3.1 Context of the Study, Research Question, and Methods

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 a convenience sample consisting of 9 students volunteering from a class of 50, enrolled in the course in 2020. The students had varied knowledge background in mathematics, but no experience with CT. The course introduced the basic constructs of the MATLAB programming language, e.g., single variables, arrays, control flow statements and functions. The course also discussed major steps in systems development, i.e., analysis, design, implementation, and testing. Ultimately programming was used for numerical analysis, and the concept of CT was briefly introduced through a worked example.

The research question addressed in this paper is: How do students engage in CT, MT and programming activities when participating in group work for solving a mathematical problem in a first-year undergraduate course?

The main data collection method is participant observation of three groups, each consisting of three students with varying knowledge in mathematics. The students were presented with a mathematical task to solve, while responding to questions in dialogue with the teacher on the solving process, by means of CT, MT, and programming activities. Open-ended questions were also used to gain a deeper understanding of the process.

The analysis of the results seeks indications of students' engagement in CT and MT when solving mathematical tasks. It uses a deductive-inductive strategy based on the interplay between the theoretical framework and the empirical data (Patton, 2002).

3.2 The Task

The task for the group work was: "Write a MATLAB-function calculating the circumference of a triangle, based on the coordinates of its three corners." This is feasible by calculating the length of each side of the triangle and adding the results, a task which may be considered to be composed of a mathematical and a computational part. The mathematical part consists of calculating the length of a triangle side based on the coordinates of the corners. This may be done by using the Pythagorean theorem, i.e., given two corners, $C_1: (x_1, y_1)$ and $C_2: (x_2, y_2)$, employing the distance formula $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, as illustrated in figure 2.

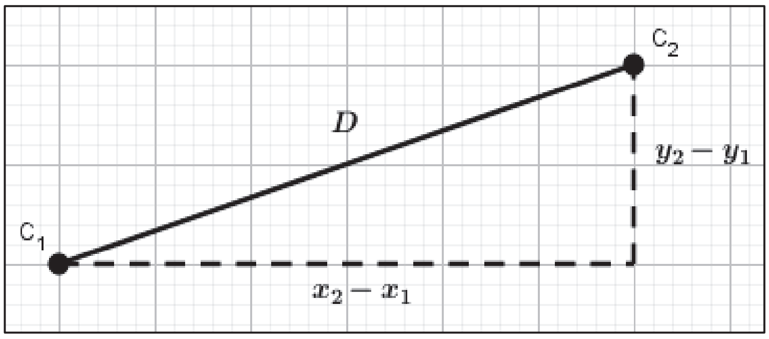

Figure 2. Employing the Pythagorean distance formula

The computational part consists of formulating an algorithm for systematically applying the distance formula to the triangle sides and adding the results. This involves creating a control flow structure for repeated use of the distance formula, i.e., a loop, as well as a selecting a convenient data structure for representing the coordinates of the corners.

3.3 Group Work Activities

The group work consisted of three sessions with three students in each, scheduled for 45 minutes. It took place in the video conferencing system Zoom (https://zoom.us/), due to corona restrictions. The sessions were recorded and transcribed. The task was introduced in the Zoom chat at the beginning of each session, and the students were asked to reflect on it and engage in a discussion on the solving process. When required, the teacher outlined steps, asked questions, and gave hints. In all groups one of the students undertook the task of coding in MATLAB, sharing screen with the others.

4. RESULTS

The results describe how the participating students engaged in solving the task, with focus on the mechanisms outlined in figure 1. During the work, the teacher suggested test coordinates based on the triangle shown in the GeoGebra screenshot in figure 3. In the following the teacher is referred to as T, and the students as S1 – S9.

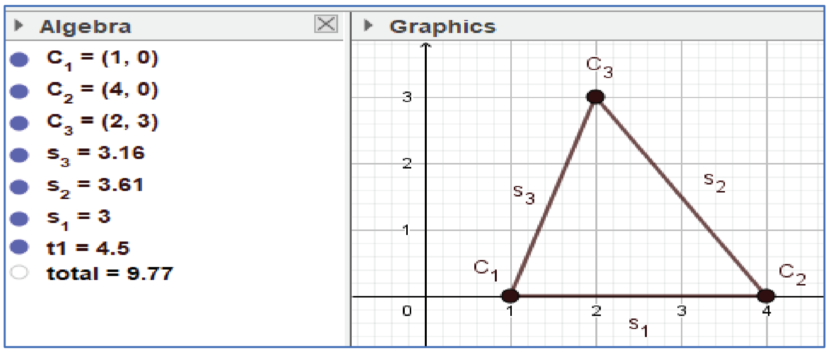

Figure 3. Triangle test coordinates

During work the students faced the problem of the end points of the sides of a triangle being numbered modularly, i.e., 1-2-3-1, whereas the loop variable progresses linearly, i.e., 1-2-3-4. A calculation modulo 3 is a good MT solution to the problem, but because MATLAB array indexing starts at 1 instead of 0, implementing the calculation in program code is a bit awkward. An alternative solution is adding a dummy point 4 with coordinates equal to point 1 to the test data.

4.1 Group 1

T presents the task, and when there is no response, prompts the students for a suggestion on how to attack it.

After a minute S1 suggests calculating the distance between the corners and then adding them. But without a clear distinction between the tasks involved. When T wants to know what to start with, S1 suggests finding a length, though with imprecise wording. At this stage, the students struggle with the initial process of understanding what the mathematical problem consists of, and they are unable to use MT and CT to model and decompose the task.

None of the students are able to suggest a method for calculating the length. When T mentions Pythagoras, S1 argues that the triangle then has to be right-angled, confusing the interior angles of the triangle with the angle to be constructed on each side, as in figure 2. Considerable elaborations from T are required to correct the misunderstanding. This shows that though the Pythagorean theorem supposedly is well known to the students, they are unable to use MT to employ it in practice without lengthy instructions.

Once the idea of calculating the length of a triangle side by Pythagoras is established, an algorithm for doing the calculation may be formulated. However, there is no student initiative to take the process further. T then starts a discussion on what type of variable that would be best suited for the coordinates, thus taking a step into the process of producing program code. S2 and S3 engage in the discussion, and after some hints from T agree to that using an array will facilitate the use of a loop. At this point S2 and S3 show that they now are able to outline the major steps in an overall algorithm:

T: (…) if we have a table, we can have a loop variable, and then we can traverse that table coordinate by coordinate.

S3: Yes.

S2: Yes, and then we do Pythagoras.

T: Yes.

S2: Step by step.

Responding to a question from T on whether it would be easier to generalize the algorithm to a polygon with an arbitrary number of corners, S2 again demonstrates to have grasped the general idea:

S2; When we are using a loop, all that is needed is adding some numbers to the array. Almost.

The students now describe the structure of an algorithm, but it is vague in detail, unsuitable for programming and testing. To continue there is a need to reduce the complexity of the algorithm, typically deciding on starting by calculating the length of one side only. T attempts to discuss this with the students, but eventually needs to suggest the solution:

T: (…) in the first version of the program, would you try to include everything, or would you construct it bit by bit?

ISBN: 978-989-8704-33-7 © 2021

S2: Eh, I would have tried to include all at one time. And then checked if it worked or not. And then, after that, if it works, I would have tried to find points where it could be improved.

T: Yes. Do you think a good strategy would be to calculate the length of a single side and make that work first, and then extend it to the triangle?

S2: Yes, that would have been a good idea.

S2 now suggests starting writing program code, and shares screen with MATLAB programming environment. On instructions from T S2 creates arrays with test coordinates based on the triangle in figure 3. With some assistance from T S2 creates a syntactically correct MATLAB function skeleton, but then work stops. T reminds of the suggestion made earlier:

T: Well, what do you feel the next step should be? We did agree not to do everything at the same time?

Still after this allusion to how to simplify the algorithm, there is no response. The students seem unable to perform the transition between algorithm and program code. T suggests calculating the change in x-coordinate between corner one and two. S2 then types "xdiff = x2 - x1". By this S2 appears to have switched to a pure MT-context because, though mathematically sound, the variable names x2 and x1 do not exist in the MATLAB function. S3 appears to see the difficulty, and is able to, assisted by T, instruct S2 on replacing x1 and x2 with the proper array variables. The students are however not able to translate the mathematical distance formula into MATLAB-code without strong hints from T.

The code is now tested, and MATLAB correctly outputs s_1 as in figure 3. When T suggests testing with corners 2 and 3 instead, S2 is able to replace the array indexes and test again. The test is successful.

Now an entire cycle in figure 1 is completed. MT has been used to determine that Pythagoras may be used to calculate the distance between two corners, CT has been used in producing an algorithm, the algorithm has been implemented in MATLAB-code, the code has been tested, and the output has been verified to make sense mathematically, in form of the length of triangle sides.

T now wants the students to engage in extending the program to calculate the circumference of the entire triangle, but work does not progress, not even when T reminds of the consensus on using a loop. T has to provide detailed instructions on how to merge the distance formula with the loop. But then S3 realizes that the loop variable, n, must be used to identify the triangle corners, and S1 employs MT to establish that if n is a natural number, the subsequent natural number is $n + 1$, and uses CT to apply this to two succeeding corners.

S3 now states that there will be a range error when $n = 3$. T postpones the problem by suggesting setting the upper loop boundary to $n = 2$.

To make the students attack the problem of making the algorithm accumulate the side lengths, strong engagement from T is again required. But finally, S3 is able to instruct S2 on how to implement the mechanism required. The program is tested, and correctly outputs $s_1 + s_2$ as in figure 3.

When T brings up the problem of the index becoming 4, S1 again employs MT and suggests modulo calculations. T acknowledges the idea but suggests using CT and adding a test corner 4 with equal to test corner 1. The program is tested and correctly outputs the circumference of the triangle, $s_1 + s_2 + s_3$, as in figure 3.

When T ends the session by asking the students to sum up the steps taken in creating the final MATLAB program, there is no response. They thus seem unable to articulate the process of using MT and CT in problem solving.

4.2 Group 2

At the start of the group work S6 suggests Pythagoras as a method to calculate the length of the triangle sides. T suggests tables as variable types for the coordinates. There is some confusion on how to provide test data, so T asks if somebody can start MATLAB. By doing this T goes straight to the programming phase, without prior elaboration on MT and CT.

S4 starts MATLAB, sharing screen. When T invites the students to suggest test data, S6 gives an odd comment:

S6: (…) We probably want some coordinates that are equal, some that are negative, some that are positive.

One may suspect that S6 here mechanically refers to a test strategy appropriate in the context of previous exercises, without abstracting, generalizing, and adapting the strategy to the context in question.

T provides the test coordinates from figure 3. S6 and S4 now engage in a discussion, and are able to create a function skeleton with only a few hints from T.

T then reverts to MT and demonstrates on figure 2 how the length of a triangle side may be calculated from the corner coordinates.

The students are unable to give a reasonable suggestion on how to split the task into subtasks, and T suggests starting by calculating a single side length. Now the students will need to use CT to implement this calculation. S6 is able to instruct S4 on what to type with very little intervention from T. The result is correct apart from that the length calculated is not assigned to the function's return variable. S4 is however able to correct the problem when T draws attention to MATLAB warnings.

The program is tested and correctly outputs the length of s_1 as in figure 3. Now T asks the students what the next step should be. S6 responds:

S6: We should probably do that for the rest of … find an expression then… probably a loop doing this for the remaining sides.

T: Yes, why do you suggest a loop?

S6: Because then you can take from point one to point three, or side one to side three.

T: Yes.

S6: A for-loop then, since you know the fixed values and the fixed steps between. That performs that operation three times, so that you do not have to write that operation for the three different points.

Though the students initially were unable to suggest loop as a control flow structure, S6 is now able to describe it in detail. Thus, after producing some program code, S6 reverts to CT and is able to extend the algorithm.

With only a few hints from T S6 instructs S4 on how to program the loop. A transition to MT/CT is then required to employ the loop variable, n, in indexing the corners. When T suggests replacing index 2 with $n + 1$, S4 is able to replace all four indexes correctly without further instructions.

The code is tested, and MATLAB outputs an array-out-of-bounds message. S6 immediately identifies the problem as $n + 1 = 4$ when $n = 3$, but suggests correcting it by setting $n = 2$ as upper loop bound, apparently unaware that this will cause the third triangle side to be skipped. T however allows the correction.

MATLAB now outputs the length of s_2 as in figure 3. When T demonstrates on figure 3 that s_1 and s_2 are not added, S6 is able to use CT and describe the algorithm extension required.

MATLAB now outputs the length of $s_1 + s_2$ as in figure 3. To include s_3, S6 first reverts to CT and suggests adding an extra instance of the distance formula. But when T relates the indexes to the triangle sides, S6 employs MT and suggests using a modulus calculation. T accepts this as a good strategy but suggests adding a corner 4 equal to corner 1, as a quick fix. MATLAB now outputs the correct circumference of the triangle as in figure 3.

When summing up S6 is able to describe how to extend the function to work on an arbitrary polygon, and also to articulate the steps undertaken in the task just completed. S6 is however not quite able to articulate a test strategy.

4.3 Group 3

When the group initially is asked to reflect on how to attack the task, S7 correctly suggests calculating the distances between the corner points and adding them, though incorrectly referring to the distances as "absolute values". T comments that S7 has mentioned two subtasks, but none of the students seem able to use CT in a systematic way to identify the tasks, even after hints. And only after a 3-minute demonstration from T on figure 2 does S9 suggest using Pythagoras to calculate a side length.

Also, even after strong hints from T, the students are unable to suggest neither test strategy nor a suitable data structure for test coordinates. Temporary single variable test coordinates are then created for corner 1 and 2.

S9 starts programming in MATLAB, sharing screen, and is able to establish a syntactically correct function skeleton, but work progresses no further. When T suggests calculating change in x-value, S9 makes the same mistake as S2 in group 1, using non-existent variable names. T is required to point out the problem, but S9 then is able to make corrections. After T uses MATLAB warnings to point out that the calculated distance is not assigned to the return variable, S9 corrects this also, and a test run outputs the length of s_1 as in figure 3.

ISBN: 978-989-8704-33-7 © 2021

T now invites the students to consider a more suitable data type for the coordinates. S7 and S9 realizes there will be a problem "to get in all the coordinates" but have no further suggestions. When T suggests arrays, S9 is however able to, with hints from T, implement the necessary changes in the MATLAB-code. S9 now raises the question of the sequence of the two corners used to calculate dx and dy:

S9: Is it number two minus number one, or what?

T: Does it really matter? Does anybody have an opinion on that? What would have happened if we swapped 1 and 2?

S7: We would have gotten another number.

S9: It would be negative.

T: Yes. We would have gotten opposite sign.

S9: Minus two. Yes.

The discussion continues, but the students are unable to employ MT to articulate that swapping two corners simply causes a change in sign, which is unimportant since the value later is squared.

Work stops. On request from T on a natural next step, S7 suggests adding the lengths, but there is no further progress. Even when T asks the students to reflect on mechanisms used in previous exercises, nobody realizes the need for a loop. T is required to give detailed instructions.

S9 seems to understand that the loop variable n will have to be used in identifying the corners, and after a hint from T is able to use MT and CT to make the adaptations required.

Now the problem with the loop variable being out of bounds arises. S9 is able to identify the problem, but there is no switch to MT to come up with modulo calculation. T then suggests adding a corner 4 equal to corner 1 to the test coordinate array.

Now a test run outputs s_3 as in figure 3. Referring to the figure, T explains what has been calculated, and what should have been calculated, hinting that the same kind of problem has been solved in previous exercises. S9 realizes that a mechanism for accumulating the side lengths is required but is unable to implement it. S8 suggests employing a variable named "sum-of-series". In the context of many previous exercises this has been a sensible name, but it absolutely does not fit the current context. S8 thus demonstrates a lack of ability to abstract, generalize and adapt a mechanism used in other settings.

Eventually T has to instruct S9 on how to implement the mechanism for adding the side lengths. The program is tested and outputs the correct circumference of the triangle in figure 3.

Ultimately T asks the students how they would attack the problem on their own. S9 would have thought about it a bit, then started to program. When asked about taking all at once, says to go by it part by part, but is unable to identify the parts. When T asks how the solution is generalizable to arbitrary polygons, S9 is able to provide an adequate answer. This demonstrates a certain ability to used CT in solving a task, but the ability has major flaws.

5. DISCUSSION AND IMPLICATIONS

The main findings of this study are twofold. Firstly, the introduction of CT to students at the undergraduate level presents many challenges for teachers committed to improving students' MT. Secondly, the mathematical task presented to the students to develop their MT and CT skills was challenging for novice learners for many reasons, considering their minimum prior knowledge of CT and programming, their varied mathematical knowledge levels, and mathematical problem-solving skills. This study produced interesting findings and implications for further research, which can be summarized as follows according to the categories and key principles of the theoretical framework.

Mathematical problem, MT-CT interactions: Most students struggled with the initial process of understanding what the mathematical problem task consisted of, and as a result were unable to use MT and CT to model and decompose the task without the teacher's guidance. It appears from the students' responses that the lack of MT hindered them in making sense of the task and develop a problem-solving strategy translatable into an algorithm. The transition between MT and CT was also challenging, even though these thinking processes are based on logical reasoning, as stated in the theoretical framework. As a result, the interaction between MT and CT was not straightforward and at times very challenging and incoherent. Most students were unable to identify the problem until the teacher pointed it out. Guided by the teacher, some students were able to translate the identified mathematical expression into correct MATLAB-code, but without the mediation of CT.

Transition algorithm – code: As a consequence of the lack of MT, most students failed to use CT to develop an algorithm step-by-step, or presented a solution only as program code without an explicit algorithm description and associated steps to be taken in obtaining a solution. Some students were able to use CT and describe an algorithm and program code. However, they often switched rapidly to the programming phase, without further elaboration on creating a coherent algorithm. The transition between algorithm and MATLAB-code was thus challenging as the students struggled with implementing the algorithm in the form of MATLAB code.

CT – algorithm-decomposition: When students managed to develop some sort of algorithm, they were often unable to reduce the complexity of the algorithm or simplify it and re-construct it step-by-step or decompose the task in smaller sub-tasks. Likewise, the direct transition from MT to code without the mediation of CT did not work well, e.g., the translation of the mathematical distance formula into MATLAB-code.

Program code: When it comes to programming, many students were able to understand the program after it had been developed under the guidance of the teacher. However, extending the program based on extensions to the algorithm proved difficult. For example, once the need for a loop was established, the students struggled with integrating the loop with the previously coded distance formula. It is worth noting however, that in one case a student, initially unable to see the need for a loop, after producing some preliminary program code was able to revert to CT and describe the mechanism required in detail.

Abstraction – Generalization: The results also show that generalization was difficult, as the students showed low ability to abstract, generalize and re-use programming concepts known from previous exercises in another context. For instance, the students needed strong guidance to see the need for a loop, and in some cases suggested reusing mechanisms unmodified from previous exercises. Likewise, few students were able to generalize the algorithm or extend the program to work on arbitrary polygons.

Code evaluation: Finally, students struggled with code validation and testing. On request from the teacher, most students were initially not able to suggest a test strategy, and ultimately, they were unable to describe the test strategy actually employed during the group work.

Implications from the results indicate that a requirement for engaging in mathematical problem-solving through CT and programming is a solid foundation in both MT and CT, as well as an understanding of the interaction between the two. More precisely, to make MT interact better with CT, a learning environment around first-year undergraduate mathematics courses should be well designed to ensure a smooth integration of CT and MT. Moreover, the learning environment should promote explicit CT processes that expose students to logical and algorithmic thinking, decomposition, generalization, abstract reasoning, and evaluation. This is the key in deciding how to introduce CT at the undergraduate level and assist students in learning to think computationally. A learning environment where students engage in CT associated with appropriate learning activities and varied and intrinsically motivating tasks that are suited to their mathematical knowledge level may lead to greater involvement and learning progression, and hopefully improved learning outcomes as well. Such a learning environment is powerful in providing more opportunities for students to develop their own understanding of CT and programming concepts. In other words, it lays the ground for learning autonomy. On the other hand, student autonomy cannot be fully expected for novices without good knowledge background in MT and familiarities with CT and programming. Moreover, as this study shows, the role of a knowledgeable teacher in MT and CT is still important to assist students in designing algorithms and implementing computational solutions for mathematical problem-solving. In this regard, there will be a need for professional development for teachers that enhances not only their understanding of CT, but also about the ways in which CT interacts with MT. A final implication of this research is the need to reconsider the interactions between MT, CT, and programming because the findings show more overlaps than what was proposed in the model in figure 1. Indeed, a more in-depth examination of the interactions between CT and MT is necessary to highlight their communalities and potential differences and suggest some modifications to the proposed model.

6. LIMITATIONS AND FURTHER WORK

The number of participants (*N=9*) of out of 50 students from one class is limited to generalize the results. Hence, a larger number of participants from one or several classes could have been more appropriate to achieve higher generalization. Nevertheless, the data analysis method used in this article to generate a large and in-depth set of qualitative data seems to be justified for addressing the research question and associated issues on CT, MT, and programming critically.

Future work will include a study exploring the implications of a learning environment with explicit focus on CT and MT in a larger number of groups to ensure more reliability and validity of the results.

REFERENCES

Buteau, C., et al. (2018). Computational thinking in university mathematics education: A theoretical framework. In A. Weinberg, et al. (Eds.), *Proceedings of the 21st annual conference on research in undergraduate mathematics education* (pp. 1171–1179). San Diego, CA: RUME.

Csizmadia, A. et al. (2015). Computational thinking: A guide for teachers. Retrieved from https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf

Filho, P., & Mercat, C. (2018). Teaching computational thinking in classroom environments: A case for unplugged scenario. In V. Gitirana, et al. (Eds.). *Proceedings of Re(s)sources 2018 - Understanding Teachers' Work Through Their Interactions with Resources for Teaching* (pp. 296-299). Lyon: France.

Hadjerrouit, S. & N.-K. Hansen (2020). Students engaging in mathematical problem-solving through computational thinking and programming activities: A synthesis of two opposite experiences. *Proceedings of the 17th international conference on cognition and exploratory learning in the digital age (CELDA 2020),* pp. 91-98.

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 CERME 9* (pp. 2524-2530). Prague, Czech Republic.

Papert, S., & Harel, I. (1991). *Constructionism*. Norwood, NJ: Ablex Publishing.

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

Shodiev, H. (2015). Computational thinking and simulation in teaching science and mathematics. In M. G. Cojocaru et al. (eds.). *Interdisciplinary Topics in Applied Mathematics, Modeling and Computational Science.* Springer Proceedings in Mathematics & Statistics 117, pp. 405-410

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.

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

Wing, J.M. (2006). Computational thinking. *Commun. ACM 49*(3), pp. 33–35.

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

Wing, J. M. (2014, January 9). 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