Chapter # - will be assigend by editors

ANALYZING STUDENTS' COMPUTATIONAL THINKING AND PROGRAMMING SKILLS FOR MATHEMATICAL PROBLEM SOLVING

A Case Study

Nils Kristian Hansen – University of Agder, Kristiansand, Norway
Said Hadjerrouit – University of Agder, Kristiansand, Norway

Abstract: The purpose of this chapter is to analyze students' computational thinking and programming skills for mathematical problem solving in a programming course at the undergraduate level. The chapter critically reviews the research literature in the field and proposes a model that connects mathematical thinking to computational thinking and programming. The model is then used as a theoretical basis to collect and analyze data from three groups of three students solving a mathematical task. The results reveal that the participating students faced several challenges: Lack of mathematical thinking, insufficient experience with computational thinking, and, more importantly, lack of deeper connection between computational and mathematical thinking, severely impeded their work. Conclusions, limitations, practical implications, and future research work are drawn from the results to explore mathematical problem solving through computational thinking and programming in a larger group of students. Studies involving a larger number of participants would be relevant to compare with findings of the present study to achieve more reliability and validity.

Key words: Algorithm, computational thinking (CT), mathematical thinking (MT), mathematical problem solving, programming.

1. INTRODUCTION

There is an increasing attention on CT and programming in mathematics education at the undergraduate level in science studies. A primary motivation for introducing CT into mathematics classrooms is the rapidly changing nature of competencies of the discipline required for future work in society and the

professional world. CT and programming can be employed in a variety of ways that reflect the specificity of the different academic disciplines. The benefit of CT and programming is that it could extend the skills characterizing MT by restructuring both how problems are formulated, modelized and solved (Kallia et al., 2021). Hence, there is a need to explore the way in which MT connects to CT and programming for solving mathematical problems.

This chapter initially introduces and critically comments on three core components that characterize the up-to-date review of the research literature on the topic, and includes definitions, concepts, and examples of current research. These core issues are mathematical thinking (MT), computational thinking (CT), and programming. Followingly, the chapter reconceptualizes the current research in the field, and presents an integration of the research domain by integrating the three core components in a single model functioning as a whole. The model is then used as a theoretical basis coupled with an appropriate methodology to understand, analyze, and critically reflect on students' abilities for mathematical and computational thinking and programming skills for mathematical problem solving. Finally, the results are discussed, practical implications and future potential of the research are drawn.

The study addresses the following research question: *How do student groups engage in mathematical problem solving by means of MT, CT, and programming in a first-year undergraduate course?*

The chapter is structured as follows. Firstly, it reviews the current literature in the field, and proposes the theoretical basis of the study. Secondly, the methodology is presented. Thirdly, the results are analyzed and discussed. Finally, the limitations and implication of the study and future work conclude the chapter.

2. LITERATURE REVIEW AND THEORETICAL BASIS

Three core components characterize the current research in the field: Mathematical thinking (MT), computational thinking (CT), and programming, and how these are connected to each other. These are presented and critically reviewed in three separate sections. Then, the chapter presents an integration of the three core components into a single model that functions as a theoretical basis of this study.

2.1 Mathematical thinking (MT)

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). A similar term for MT is mathematics abilities, which refers to "a combination of cognitive abilities and knowledge that contribute to one's performance on a wide range of mathematical tasks" (Blacksmith 2020, p. 2783). Accordingly, it is argued that individuals with high levels of mathematical abilities are more likely to successfully complete difficult mathematical tasks, in contrast to individuals who cannot complete easy mathematical tasks. These are described as having low levels of mathematical abilities.

MT and CT share several communalities: Mathematical modelling, problem solving, data analysis, statistics, and probability (Ang, 2021; Shute, Sun, and Asbell-Clarke, 2017; Weintrop et al., 2016). Mathematical modelling provides a good foundation for the use, practice, and development of CT. Likewise, problem solving is common both to MT and CT. Data analysis requires numerical thinking in problem solving, especially when empirical data are involved. Moreover, it is argued that CT improves logical and reasoning skills, problem solving, and academic performance in mathematics education (Martínez-García, 2021).

2.2. Computational thinking (CT)

CT is basically more about thinking than computing and computer programming (Li, Schoenfeld, & diSessa, 2020). It represents a "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use'' (Wing 2006, p. 33). Clearly, CT is a fundamental skill for virtually any discipline, including mathematics, physics, and engineering.

More specifically, 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 and 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. On the other hand, 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, 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; 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.

These cognitive processes correspond to those being accepted as the four "cornerstones" of CT, or taxonomy of CT practices (Weintrop et al. 2016). These are data practices, modelling and simulation practices, computational and problem-solving practices, and, in lesser degree, system thinking practices.

Data practices are about collecting, creating, manipulating, analyzing, and visualizing data. These practices correspond to abstraction in CT, which aims at gathering important data to carefully examine the problem by neglecting non-essential aspects, as highlighted above. *Modelling and simulation* practices are ways to implement abstraction, decomposition, generalization, and design algorithms. These practices are about constructing, using, and assessing computational models to understand a problem, find and test solutions. *Computational problem-solving* practices relate both to CT (especially abstraction and evaluation) and programming, and consist of preparing and developing computational solutions, choosing effective computational tools, assessing different approaches to problems, as well as troubleshooting and debugging. These practices are not entirely new, and are in high accordance with those observed as characteristics of MT. In addition, these practices are linked to computer programming as highlighted above.

2.3. Programming

Programming has changed over the last 30 years, but the basic principles remain the same. Programming is based on logic, procedures, and functions (Misfelts & Ejsing-Duun, 2015). According to Ang (2021), one effective way of developing CT skills and practices would be through learning programming and solving problems that require some form of coding. One problem-solving skill is to think in a logical and systematic manner, and develop an algorithm, by breaking problems into smaller parts, making use of flowcharts to visualize the flow of the process. This skill is equivalent to simplifying or decomposing a problem and constructing procedures (or functions) and sub-procedures in the code. Moreover, writing code requires following the syntax of the language being used, and keeping to certain rules, which means to practice abstraction or abstract reasoning, because only the most important and relevant pieces of information will be extracted and used, like modelling in MT.

Coding also requires the use of variables as representations of factors involved in a problem. Moreover, coding involves repeating a piece of code or iterating code through loops and managing data sets as well. Such skills help to develop a sense of pattern recognition and identify similarities and differences in tackling modelling problems, which connects well to MT. However, it is not enough to know basic programming to be able to use programming techniques in a mathematical context. Programming is closely related to CT, but traditionally, programming education required producing algorithms from scratch (Kaufmann & Stenseth, 2020). Nowadays, it is recommended to pass through CT to be able to design algorithms (Wing, 2014). *Designing algorithms* involves creating representations of the solution process, including representations such as flowcharts, pseudo-code, or systems diagrams. It involves further activities of decomposition, abstraction, and generalization. *Coding* is the next essential step to translate the algorithm into code form and evaluating it to ensure that it functions correctly under all anticipated conditions. *Debugging* is the systematic analysis and evaluation using skills such as testing, tracing, and logical thinking to predict and verify outcomes. Finally, *applying* is the adoption and generalization of pre-existing solutions to meet the requirements of a similar problem in another context. This includes the development of an algorithm and subsequent program in one context that can be re-used in a different context.

2.4. Connecting MT, CT, and programming

Connecting MT, CT, and programming in a meaningful way is at the heart of tackling mathematical problem solving. 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). As highlighted above, MT and CT have a lot of communalities such as problem solving, modelling, data analysis and interpretation. Moreover, the relationship between modelling, analysis, and solution of mathematical problems falls under the umbrella of CT and is grounded on computer programming (Buteau, et al., 2018).

The close connection between MT and CT provides opportunities for building efficient algorithms for mathematical problem solving in form of 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 smaller 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 to 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.

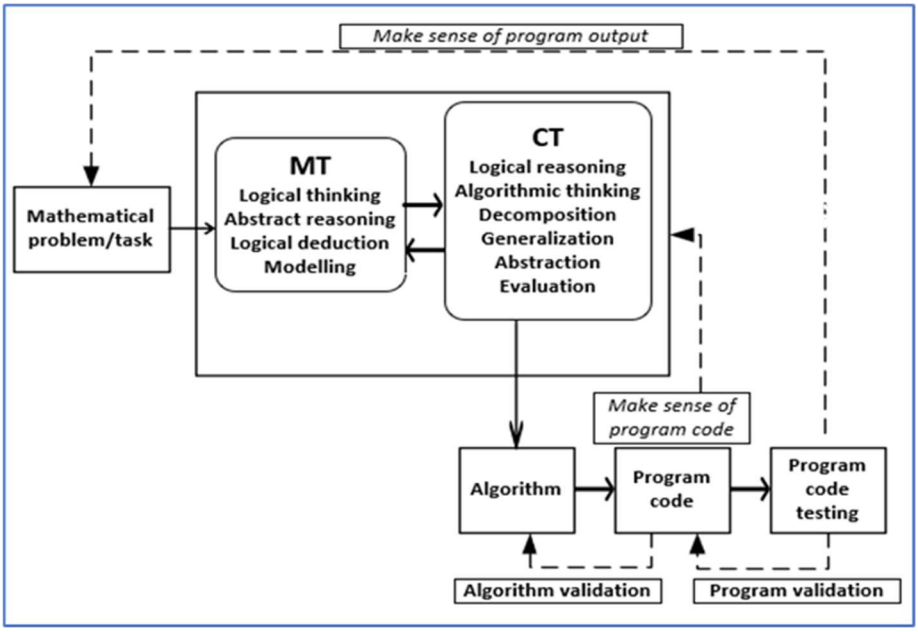

Figure 1. A model for mathematical problem-solving connecting MT, CT, and programming.

This model works as follows. 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 starting from a mathematical problem and ending 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.

Based on the model in figure 1 and the taxonomy of CT practices proposed by Weintrop et al. (2016), the goal is to gather and analyze data on students' CT, MT, and programming activities

3. METHODOLOGY

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 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 MT, CT, 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 problems. It uses a deductive-inductive strategy based on the interplay between the theoretical basis of the study 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."

A fruitful approach to solve the task is to start by using CT to build a coarse computational model. Decomposition will be an important factor in this, as the task naturally splits into two subtasks, one for calculating the length of a triangle side, and one for systematically adding the side lengths.

To determine how to calculate a side length, MT is required, as the Pythagorean theorem will be a natural choice. I.e., given two corners, C_1: (x_1, y_1) and C_2: (x_2, y_2), the corresponding side length can be calculated by the Pythagorean formula $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, as illustrated in figure 2.

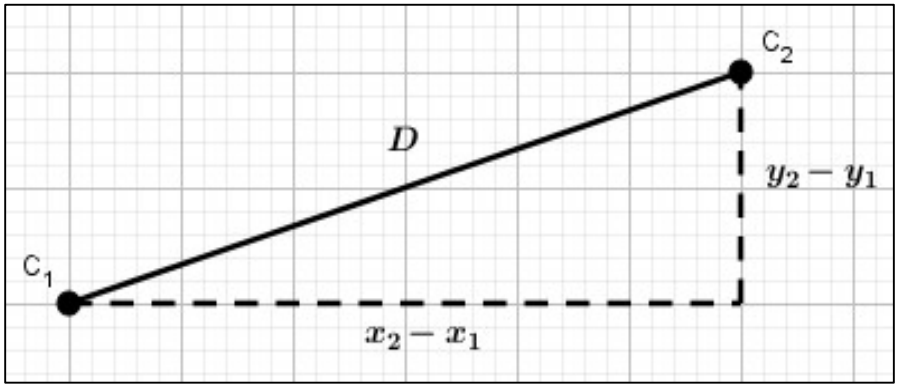

Figure 2. Calculating distance by the Pythagorean theorem.

A switch back and forth between MT and CT is then necessary, in order to create a model of how to the coordinates provided in a MATLAB function can be employed in the Pythagorean theorem. The given task does not specify any particular data structure for the coordinates, so making an appropriate choice is required. Considering the need for systematically processing the sides, an array will be appropriate.

A natural next step will be to design an algorithm for calculating the length of a side, translate the algorithm into MATLAB code, and validate it through testing. If the test fails, one must revert to CT, and reason about the cause of the error and how it can be corrected. There may be several iterations involving validation and correction until the code passes the test.

Switching to the second subtask, CT is required to create an algorithm for systematically processing the three sides, adding the lengths. A *for*-loop should present itself as a convenient control structure, easily implementable in MATLAB. Followingly a switch back and forth between CT and MT is required to associate the triangle corners with the loop variable.

At this point a border value problem must be addressed, since triangle corners are numbered modularly, i.e., 1-2-3-1, whereas loop variables progress linearly, i.e., 1-2-3-4. The problem may be remedied both by using MT and CT. An MT solution will be to calculate modulo 3. However, since MATLAB array indexing starts at 1 instead of 0, implementing this solution in program code is a bit awkward. A CT solution will be adding a dummy point 4 with coordinates equal to point 1 to the coordinate array.

Next a new program test is appropriate. Again, several loops involving correction and testing may be required until the code passes.

Finally, using CT in doing a generalization will be natural. An obvious generalization will be to expand the algorithm and program to work on polygons with an arbitrary number of 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, due to COVID-19-restrictions. The sessions were later 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 processes 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.

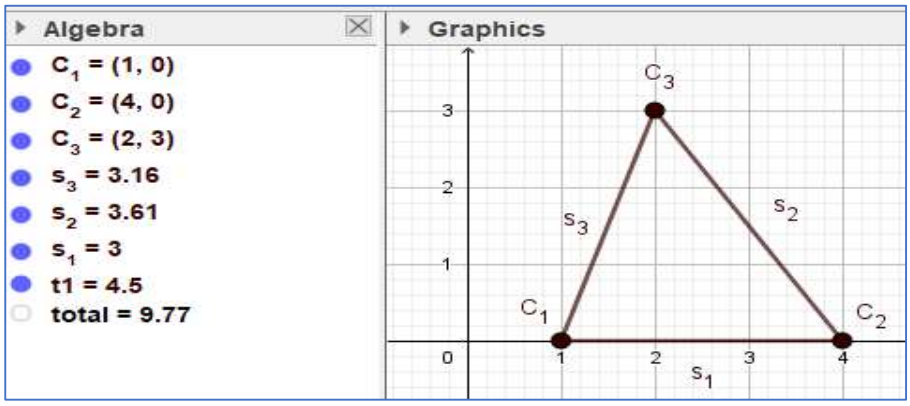

Figure 3. Triangle test coordinates

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 are unable to use MT and CT to design a coarse computational model. T then proposes establishing a method for calculating the length of a triangle side as a subtask to begin with.

It takes considerable elaborations from T to convince the students that the Pythagorean theorem may be used for calculating a side length. Though the theorem supposedly is well known, the students seem unable to use MT to employ it in practice.

Next an algorithm for the calculation may be formulated, but there is no student initiative to take the process further. T then starts a discussion on selecting a data structure appropriate for the coordinates. The students realize the need for a variable for each coordinate pair but fail to generalize the idea to a data structure convenient for handling three coordinate pairs in succession.

S2 and S3 engage in a discussion initiated by T, and after some hints agree to that an array will be appropriate.

At this point S2 and S3 demonstrates that they now are able to outline the major steps in an overall algorithm:

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

S3: Yes.

S2: Yes, and then we do Pythagoras.

T: Yes.

S2: Step by step.

T now tries to make the students realize that completing an algorithm and testing the code for calculating a side length will be a fruitful next step, but without success. T then attempts to make the students reflect on generalizing the algorithm, making it applicable to polygons. S2 now 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 followingly describe the structure of an algorithm, but it is vague in detail. T again attempts to discuss the idea of completing and testing the first subtask, but eventually is required 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?

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.

Next S2 suggests starting to write program code, and shares screen with the MATLAB programming environment. Apparently, the students are more comfortable with proceeding straight to the programming phase than theoretically elaborating on an algorithm.

On instructions from T S2 creates an array with test coordinates based on the triangle in figure 3, and with some assistance creates a syntactically correct MATLAB function skeleton, but then work stops. T reminds of the suggestion made earlier of not doing everything at once. Though, still after this allusion on how to simplify the algorithm 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". This is correct in an MT-context, but unrelated to the actual MATLAB-variables. S3 spots the difficulty, and is able to, assisted by T, instruct S2 on correcting the problem. The students are however unable to implement the Pythagorean theorem as 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, S2 is able to replace the array indexes and test again. The test is successful.

Now the subtask of calculating the length of a triangle side is finished, and 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.

T now wants the students to engage in the second subtask. Work however does not progress, 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 corner number, the adjacent corner number is $n + 1$.

S3 now identifies the border value problem that will appear when $n = 3$. T postpones the problem by suggesting setting the upper loop boundary to $n = 2$, temporarily.

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

T now instructs S2 to set the upper loop boundary back to $n = 3$. A test is run, and MATLAB responds with an index-out-of-bounds error message.

S1 now employs MT and suggests modulo calculations. T acknowledges the idea but suggests using CT and adding a fourth coordinate pair equal to

corner 1. The program is tested, and correctly outputs the circumference of the triangle, $s_1 + s_2 + s_3$, as in figure 3.

T now describes how the algorithm may be extended to work on an arbitrary polygon, without inviting the student to reflect on it.

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. But when T outlines the steps undertaken, the students agree to that the experience will be profitable when engaging in similar tasks later.

4.2 Group 2

S6 has some prior knowledge of the task and starts by suggesting the Pythagorean theorem as a method for calculating the length of the triangle sides. T then suggests starting with a single side and making that work, thus doing a decomposition in subtasks and suggesting a workflow, instead of leaving that to the students.

S6 now proposes arrays as a suitable data structure for the corner 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 further 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 current context.

T now suggests creating test data i MATLAB, based on figure 3. S6 followingly takes charge and instructs S4 on how to create a data structure in the form of MATLAB arrays.

With the test data ready, T suggests creating a MATLAB-function for calculating a side length, but there is no response. The students thus seem unable to make the transition from the MT idea of the Pythagorean theorem to actual program code. However, on hints from T, S6 again takes charge and instructs S4 on writing a function header with the coordinate arrays as arguments.

The function body is however empty, the students still have not been able to produce neither algorithm nor program code.

T then reverts to MT and demonstrates on figure 2 how a side length may be calculated from the corner coordinates. But the student's first attempt to

provide a function body reveals that they confuse the two subtasks. T reminds that the current subtask is to calculate the length of a single side.

S6 now is able to instruct S4 on what to type, with very little intervention from T. The result is correct, apart from that the calculated length is not assigned to the function's return variable. This is a flaw in implementing the algorithm as program code, which S4 is able to correct when T draws attention to MATLAB warnings.

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

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 will execute the operation three times, so that you do not have to write the 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 combine the subtasks of 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 uninstructed.

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 omission of the third triangle side. T however allows the correction.

MATLAB now outputs the length of s_2, as in figure 3. T then demonstrates on figure 3 that s_1 and s_2 are not added, hinting on a missing mechanism that has been used in previous exercises. S6 takes the point and is able to use CT to incorporate addition in the loop.

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 generalize the algorithm to work on an arbitrary polygon, in the form of MATLAB mechanisms. S6 is also able to articulate the steps undertaken in the completed task but is 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 employs MT and CT in a first thought about systematically calculating and adding the distances between adjacent corners. The thought is somewhat unclear, however, as S7 refers to distances as absolute values. T comments on 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. But when T identifies them, S7 correctly identifies calculating the length of one side as the task to start with.

The students are however unable to suggest a method for calculating a length, even when T refers to previous exercises. S7 even naively suggest calculating the length of a hypothenuse by adding the lengths of the remaining triangle sides. A 3-minute demonstration from T on figure 2 is required until S9 suggests using Pythagoras.

T now proposes creating an algorithm and a program for testing this idea. S9 starts sharing screen with MATLAB. But, even after strong hints from T, the students are unable to suggest neither test strategy nor a suitable data structure for the coordinates. To simplify, T suggests creating single variable test coordinates for corner 1 and 2. Dictation is however required for S9 to create the variables in MATLAB code.

S9 sketches a MATLAB function but without a parameter list. When T suggests calculating the change in x-value, S9 employs the symbols x1 and x2, which are nonexistent in the function. T claims that there is a problem, but nobody is able to identify it. When T explains that x1 and x2 must be provided explicitly, S9 however creates the parameter list required.

S9 is now able to translate the Pythagorean theorem into correct MATLAB code.

A test is run, but the program gives no output since the function's return variable is not assigned a value. When T draws the attention to MATLAB warnings, S9 however detects and corrects the error.

A new test outputs the correct length of s_1 as in figure 3.

T followingly invites the students to consider a more suitable data structure for the coordinates. S7 and S9 realizes there will be a problem "to get all the coordinates in" but have no further suggestions. When T proposes arrays, S9 is however able to, with hints from T, to implement the necessary changes in

the MATLAB-code. S9 now raises a question of the order of the two corners used to calculate side length:

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 determine that the order of the endpoints is unimportant when calculating the length of a line.

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 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.

The program code is tested, and the boundary value problem manifests itself. The students are however unable to suggest neither MT nor CT solution to the problem. T then instructs on adding a corner 4 equal to corner 1 to the coordinate array.

Now a test run outputs s_3 as in figure 3. Referring to the figure, T explains what the error is, hinting that the same kind of problem has been solved in previous exercises. S9 realizes that a mechanism for adding 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 is inappropriate in the current context. S8 thus demonstrates a lack of ability to abstract, generalize and adapt a mechanism from previous experiences.

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

Ultimately T asks the students how they would have attacked the problem on their own. S9 would have thought about it a bit, then started to program. When asked about taking it all on at once, says to go by it part by part, but is unable to identify the parts. When T asks how the solution may be generalized to work on 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

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. The findings report on the interactions and transitions that emerged between MT, CT, and programming that were observed during the mathematical problem-solving process. These are summarized as follows:

Mathematical problem, MT - CT interactions: Most students struggled with the initial process of understanding what mathematical problem the 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' interviews 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 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 low 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 algorithmic 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 or simplify the algorithm and re-construct it step-by-step, or decompose the task in smaller subtasks. 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 merging the previously coded distance formula with the loop. It is worth noting, however, that in one case a student, initially unable to see the need for a loop, was able to revert to CT and describe the proper mechanism after writing some preliminary program code.

Abstraction – Generalization: The results also show that generalization was difficult, as the students demonstrated low ability to abstract, generalize and re-use programming concepts known from previous exercises in new contexts. For instance, the students required 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 initially were incapable of suggesting a test strategy, and ultimately were unable to describe the strategy actually employed during the group work.

6. IMPLICATIONS

Four implications can be drawn from this study. Firstly, 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.

Secondly, 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 algorithmic thinking, decomposition, generalization, abstract reasoning, evaluation, and associated with the four “cornerstones” of CT (Weintrop et al. 2016). 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, varied and intrinsically motivating tasks that are suited to their mathematical knowledge level may lead to greater involvement and learning progression. Such a learning environment is potentially 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. However, student autonomy cannot be fully expected for novices without good knowledge background in MT and familiarities with CT and programming.

Thirdly, 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.

7. LIMITATIONS AND FURTHER WORK

The low number of participants (*N=9*) limits how much the results can be generalized. Hence, a larger number of participants from several classes would have been more appropriate to achieve higher generalization. Nevertheless, the data collection and analysis method used in this study to generate a large and in-depth set of qualitative data, seems to be justified for addressing the research question and issues on CT, MT, and programming critically. Thus, even though this study is limited to task-based interviews of three groups of three students each, the findings help to advance current knowledge in the field both from a theoretical and practical point of view.

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 participants to ensure more reliability and validity of the results.

REFERENCES

Ang, K.C. (2021). Computational thinking and mathematical modelling. In: F. K. S. Leung et al. (eds.). *Mathematical Modelling Education in East and West. International Perspectives on the Teaching and Learn* (pp.19-34). Springer Nature Switzerland AG 2021.

Blacksmith, N. (2020). Mathematical abilities. In: V. Zeigler-Hill, & T. K. Shackelford (2020). *Encyclopedia of Personality and Individual Differences* (pp. 2783-2785). Springer.

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., & Hansen, N.-K. (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.

Kallia, M., et al. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi study. *Research in Mathematics Education*, *23*(2), pp. 159-187.

Kaufmann, O.T., & Stenseth, B. (2020). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, *52*(*7*), pp. 1029-1048.

Li, Y., Schoenfeld, A.H., diSessa, A.A., et al. (2020). Computational thinking is more about thinking than computing. *Journal for STEM Education Research 3*, pp. 1–18.

Martínez-García, E. (2021). Computational thinking: The road to success in education. *Academia Letters*, Article 3973.

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 22*, pp. 142-158.

Weintrop, D. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology 25*, pp.127–147.

Wing, J. M. (2006). Computational thinking. *Communication of the 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). Computational thinking benefits society. In: *Social Issues in Computing,* January 10*,* 2014. New York: Academic Press. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html