Computational Thinking in the Mathematics Classroom: Fostering Algorithmic Thinking and Generalization Skills Using Dynamic Mathematics Software

Sylvia van Borkulo
Utrecht University
Utrecht, The Netherlands
s.vanborkulo@uu.nl

Christos Chytas
Radboud University
Nijmegen, The Netherlands
christos.chytas@ru.nl

Paul Drijvers
Utrecht University
Utrecht, The Netherlands
p.drijvers@uu.nl

Erik Barendsen
Radboud University & Open Universiteit of the Netherlands
Nijmegen, The Netherlands
erik.barendsen@ru.nl

Jos Tolboom
SLO, Netherlands Institute for Curriculum Development
Groningen, The Netherlands
j.tolboom@slo.nl

ABSTRACT

There are increasing calls for mathematics teachers to foster computational thinking (CT) skills in their lessons and align them with existing curricula and national educational policies. Algorithmic thinking (AT) and generalization are two key elements of CT that are often underrepresented in traditional mathematics lessons. This study investigated how to address AT and generalization aspects in 12th-grade calculus lessons using the dynamic mathematics software GeoGebra. We present a six-lesson intervention designed by an interdisciplinary team of researchers and teachers with a background in computer science and mathematics education that aims to foster pre-university students' AT and generalization skills in calculus lessons. We evaluated the intervention in a 15 students classroom in the Netherlands through the analysis of students' workbooks, files, interviews, and the teacher's logbook. The findings suggest that the intervention was favorably seen by both the teacher and the students, and that their learning and teaching experience was highly satisfactory. The most common challenges for successfully completing the designed material included issues related to getting familiar with GeoGebra, syntax, and effectively using conditional statements. Finally, we report on the learning and teaching experience and discuss strategies to address AT and generalization aspects for teachers who wish to address such CT aspects in mathematics lessons.

CCS CONCEPTS

• **Social and professional topics → Computational thinking**.

KEYWORDS

computational thinking, algorithmic thinking, generalization, mathematics education, GeoGebra

WiPSCE '21, October 18–20, 2021, Virtual Event, Germany

ACM ISBN 978-1-4503-8571-8/21/10.
https://doi.org/10.1145/3481312.3481319

ACM Reference Format:
Sylvia van Borkulo, Christos Chytas, Paul Drijvers, Erik Barendsen, and Jos Tolboom. 2021. Computational Thinking in the Mathematics Classroom: Fostering Algorithmic Thinking and Generalization Skills Using Dynamic Mathematics Software. In *The 16th Workshop in Primary and Secondary Computing Education (WiPSCE '21), October 18–20, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3481312.3481319

1 INTRODUCTION

Computational Thinking (CT) is an essential skill for everyone in an increasingly computing-oriented world. According to Wing [47], it should be fostered as other crucial skills like "*reading, writing and arithmetic*" (p. 33). After the popularization of Wing's influential article, there are global efforts to foster CT in primary and secondary education, in STEAM (Science, Technology, Engineering, Arts, Mathematics) and humanities. In more than a decade, CT research is evolving and focuses on specific CT elements that include abstraction, decomposition, pattern recognition, algorithmic thinking, and generalization among other aspects.

In technology-rich mathematics education, computational tools as a means of both learning and problem-solving offer promising opportunities to foster CT within learning activities that are aligned with existing curricula and educational policies. However, embedding CT thinking elements and practices into mathematics lessons is a complex and challenging process that needs to be tested and refined before infiltrating schools' curricula to prepare teachers for the upcoming challenges that inevitably occur.

This work investigated how to address AT and generalization aspects in 12th-grade calculus lessons using the dynamic mathematics software (DMS), *GeoGebra* [24]. We present a six-lesson intervention designed by an interdisciplinary team of researchers and teachers with a background in computer science and mathematics education that aimed at fostering CT in pure mathematics lessons through the use of GeoGebra. Even though the designed intervention touches on various aspects of CT, we focus on AT and generalization, two key CT elements that are often missing in traditional mathematics lessons.

We implemented and evaluated the designed intervention for pre-university students in a calculus class in the Netherlands to answer the following research questions:

- **RQ1** : How can AT and generalization aspects be addressed using DMS in 12th-grade calculus lessons?
- **RQ2**: What challenges do students encounter in successfully completing AT and generalization tasks using DMS in 12th-grade calculus lessons?

To answer the research questions at hand, we used a mixed-methods approach which we discuss in section 3. We hope that this study will contribute to existing efforts for designing computationally rich mathematics lessons that touch various aspects of CT. This study aims at informing educators about the challenges that will inevitably occur while addressing AT and generalization aspects in calculus lessons.

2 THEORETICAL BACKGROUND

2.1 Computational Thinking

Computational thinking (CT) receives more and more attention from educators, educational researchers, and policymakers in an increasingly computational world. The term CT was popularized in Wing's influential article [47] where she argued that *"to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability"*. However, Papert first introduced the term [36], envisioning computer programming as a means for developing children's procedural thinking by algorithmically solving problems. Papert did not define CT, and a universal definition seems to be missing in the academic literature [23]. Wing [47] initially highlights the importance of CT with respect to the impact of computation and computer science in modern life: *"computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer scienc"* (p. 33). In later work, Cuny, Snyder and Wing [14] provide a working definition: *"Computational thinking is the thought processes involved in formulating problems and expressing its solution as transformations to information that an agent can effectively carry out"*.

Even though there is no universal definition for CT, there are universally accepted CT elements like abstraction [30], decomposition [23], AT [3], and generalization [2, 40], among other elements. In this work, we focus on AT and generalization, two underrepresented elements in CT embedded activities in mathematics education.

Assessing CT activities is a complex and challenging task because of multiple CT aspects that are at interplay, and because there is a lack of concrete definitions, as well as different contexts and focus in existing studies investigating CT [45]. The work of Brennan and Resnick [7] presents a framework for assessing CT, which includes computational concepts (e.g., loops and selection), computational practices (e.g., testing and debugging), and computational perspectives (of individuals about their perception of the world and themselves). Even though the framework is being used with respect to the educational programming language Scratch, it can also be used to assess computational activities in different programming environments.

Studies focusing on specific CT aspects can benefit from a working model for *"operationalizing"* CT. Atmatzidou and Demetriadis [3] use a model that includes five main aspects of CT, namely *"abstraction, generalization, algorithm, modularity and decomposition"* (p. 661) and various data collection methods, including *pre* and *post-tests*, *interviews*, a *think-aloud protocol*, *observations*, and *opinion questionnaires*.

2.2 Algorithmic Thinking and Generalization

Algorithmic thinking is a crucial component of computer science [19] and CT [47]. It includes a wide range of thinking skills for understanding and designing algorithms, including *"the ability to construct a correct algorithm to a given problem using the basic actions"* and *"the ability to think about all possible special and normal cases of a problem"* (p. 160) [19].

The design and construction of an algorithm can be implemented using logical structures like sequence (a series of steps executed in a specific order), selection (a logical construct where a section of code is executed when a condition is met), and loop (a structure that repeats a sequence of instructions until a specific condition is met). In computationally rich activities, several CT elements are at interplay. For example, AT is required to consider special and regular cases of a problem when finding a generalized solution to the problem.

Generalization is also related to abstraction in the sense that regular cases are formulated as general notions. Abstraction is considered a key element of CT, and Wing [47] refers to it as the *"mental tool of computing"*. However, the concepts of abstraction and generalization are vital in mathematics [33] and have been receiving focus in mathematics research for decades now. Even though there are differences between CT aspects like abstraction in mathematics and abstraction in computer programming [47], there are parallels between CT and mathematical thinking that are receiving increasing attention in order to address CT aspects in mathematics [26] and science [44].

2.3 CT in Mathematics Education

After a relatively stable period of using graphing calculators, recent developments in mathematics education include the use of tablets and laptops, interactive whiteboards, and online interactive environments for learning and assessment. Despite the widely recognized affordances of such tools, their productive exploitation for teaching is complex [32]. Furthermore, using sophisticated computational tools in the mathematics classroom requires the appropriate techniques for using them and insights into what is happening and what should be done [16]. However, the use of such sophisticated tools in the classroom does not necessarily lead to the cultivation of CT. Computational thinking-embedded activities should be aligned with suitable didactical approaches and pedagogies [3, 20].

Initiatives have been taken to explore and research the integration of CT in mathematics education, in K-12 and in higher education. The topics addressed show large diversity, from group theory in grade 3-6 [21], geometry in grade 8-10 [28], to probability and statistics [13, 17], and use a wide range of tools, such as Scratch [10, 15], Logo-based tools [28], spreadsheets [29, 39] and the graphical calculator [27]. However, studies related to secondary education calculus lessons and CT are underrepresented. This field is worthwhile to explore in more depth and the relation of CT and

mathematics is also acknowledged in the OECD's new Programme for International Student Assessment survey (PISA) mathematics framework. [34]. This framework argues that mathematical literacy in the 21st century includes mathematical reasoning and aspects of CT.

As informatics is still not a mandatory subject in K-12, there are limited opportunities for fostering algorithmic design in the classroom [43]. However, mathematics and science frequently require the use of computational tools, and they provide rich opportunities for fostering CT [25, 46]. Moreover, as mathematics and science are more represented in school curricula, there are greater chances of fostering CT - a fundamental skill for 21st-century citizenship - for historically underrepresented populations [44].

3 METHODS

We designed, implemented and evaluated CT-embedded calculus learning activities focusing on AT and generalization. This exploratory case study took the form of an educational design research (EDR) approach [42, 49]. Below, we present the intervention design and the educational context of the developed activities.

3.1 Intervention Design

3.1.1 Educational Context. The intervention took place physically at a middle size secondary school (about 900 students) in a middle size village (about 30.000 inhabitants) in the Netherlands. The intervention consisted of six calculus lessons of 45 minutes each and took place in mandatory mathematics lessons. By the end of the learning activity, students were expected to be able to use parameters, conditional statements, and iterations in the *GeoGebra* environment (http://geogebra.org) to tackle problems in calculus-related assignments.

3.1.2 Learning Activity. The learning activity is a part of a more extensive study looking at students' learning outcomes in CT in pure and applied mathematics. It was designed by an interdisciplinary team of researchers and teachers with a background in computer science and mathematics education. The mathematics content (e.g., perpendicular bisector, focal points, tangents, zeros of functions) and aspects (e.g., object formation, encapsulation) were addressed in the learning activity under the lens of AT (e.g., the use of logical structures such as sequences, selections, and loops) and generalization (e.g., with the use of variables for finding a general solution to a mathematical problem).

The teacher led the learning activity by introducing the designed learning material to the students and helping them to tackle problems when needed. He was familiar with the used software GeoGebra and was involved in the co-design of the lesson series.

3.1.3 Learning Materials and Tools. The activity included using a workbook with tasks on paper and the open-source computer program GeoGebra. Completing the designed assignments required a good synergy of AT and generalization skills, and also using computational concepts such as conditional statements, iterations, and parameters.

The workbook consisted of seven chapters that guided the students through functional algorithms in the GeoGebra environment.

In Chapter 1 to 4, conditional statements were used. Examples of using conditional statements included the calculation of the slope of a line through points A and B, considering the case that the line might be vertical, i.e., points A and B are directly above each other. Another example is using a conditional statement so that the perpendicular bisector of line segment AB is drawn in the exception case of a horizontal line.

Chapter 5 and 6 used iterations, for example, to create a specific number of tangent lines. Chapter 7 was a (voluntary) final task about the Newton-Raphson method for calculating zeros of a function and using iterations and a macro (a sequence of computing instructions). For Chapter 7, the students could voluntarily make an end report.

All chapters required generalizing from a specific case to a more general one by using parameters to create a general solution. For example, creating a general solution by defining an equation that depends on the coordinates of points A and B, so that when a student drags the points A and B, the equation changes along with them. All assignments required the combined use of generalization and algorithmic thinking skills.

GeoGebra is dynamic mathematics software that is used for geometry, algebra, statistics and calculus lessons in primary, secondary and higher education and can be downloaded or used online for free on geogebra.org. It allows the creation of points, lines, segments, vectors, among other mathematical representations, which can be dynamically altered once instantiated. GeoGebra can store variables for mathematical objects such as numbers, points, line segments, vectors, equations and fuctions. Objects can be entered and modified by using buttons or by using input fields. Moreover, GeoGebra allows the use of iteration lists and conditions and can be used to introduce computational concepts in computing education. In GeoGebra, generalization is practiced mainly through the use of variables, which can be combined with conditional statements. An example is when using variables in a general solution of a problem that also includes special cases, e.g., the denominator of a fraction in an equation being zero. One valuable feature of GeoGebra is its visualization possibilities which enhances the exploration of mathematics [1].

3.2 Participants

Our study sample consisted of 15 twelfth-grade pre-university students (eleven female and four male) that were 17-18 years old. The students did not have prior knowledge of the GeoGebra software, but some had seen their teacher using it in previous lessons. The students were informed about the study's aims and could voluntarily consent to use their data to evaluate their learning experience. We took ethical considerations into account and put great efforts into ensuring the implementation of good research practices for underage populations. Eleven of the students have provided us with complete data.

3.3 Data Collection

3.3.1 Workbooks and GeoGebra Files. We collected workbooks and GeoGebra files of the students in order to identify how the students tackled AT and generalization tasks and what challenges they encountered in successfully completing them.

3.3.2 Interviews. To better understand students' learning experience with the designed learning activities, we conducted semi-structured interviews with students who were willing to participate. In total, 12 students participated in the interviews, which took place during the course of the lesson series.

We asked students about their opinions on the learning activities, how they solved the tasks and how they implemented CT concepts and practices. All interviews were transcribed and were analyzed using the developed codebook.

3.3.3 Teacher's Logbook. To investigate the learning and teaching experience of the educational experiment from the teacher's perspective, we asked the teacher to fill in a logbook and take notes regarding the teaching and learning experience.

The logbook manuscripts were analyzed using the codebook. Additionally, the focus was on student understanding and learning outcomes, and planning and instruction

3.4 Data Analysis

To tackle the research questions at hand, we followed a mixed-methods approach to triangulate our findings. A mixed-methods approach is suitable for investigating phenomena with novel educational technologies and provides more robust evidence on the impact of the designed intervention [38].

To capture and analyze CT aspects in practice in the different data sources, we developed a codebook following a deductive and inductive approach. Based on the design of the assignments, the themes (1) *generalisation* and (2) *algorithmic thinking* with subthemes (2.1) *translation of mathematical tasks into computational steps in GeoGebra*, and (2.2) *logical structures* (such as, sequences, selections and loops) were predefined. From the interview transcripts of students, their GeoGebra files and workbooks, and the teacher's logbook additional themes were derived: (3) *students' strategies for AT and generalization*, (4) *perceived difficulty*, (5) *encountered problems or challenges*, and (6) *resources to tackle problems*.

We used data triangulation and investigator triangulation [12] and analyzed the different types of data collaboratively. The researchers worked in close cooperation and met weekly in an iterative process to make notes on manuscripts and compare codes.

We analyzed the students' successful completion rate using the students' workbooks and GeoGebra files, the students' self-reported strategies with respect to AT and generalization, respectively, as expressed in the interviews, and the teacher's experience from the teacher's logbook. Finally, we captured the encountered challenges through the analysis of the workbooks and Geogebra files as well as the interviews with the students.

4 RESULTS

4.1 Successful Completion of the Assignments

Overall, according to the teacher of the class the successful completion rates of the workbooks and GeoGebra files were highly satisfactory. For evaluating the completion rates of students' work, we used the descriptions "incomplete" for assignments that included less than half of the tasks successfully completed, "partially complete" for the assignments that included at least half of the tasks successfully completed but the rest were missing or had mistakes, and "successfully completed" for the assignments that were fully completed without mistakes. Seven students included parts of code that were unnecessary and did not serve any purpose in solving the calculus tasks, but we described them as "successfully complete" in the case that the solution was working. Two GeoGebra files for two chapters were not submitted by two of the students.

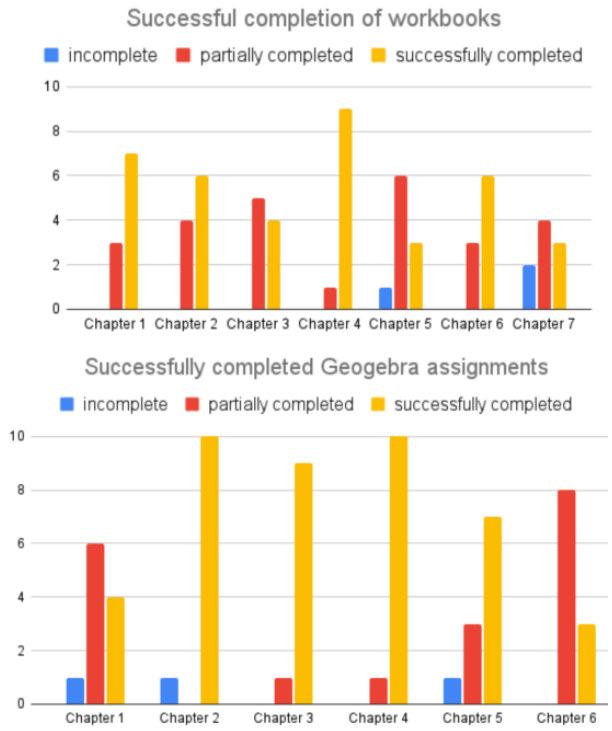

Figure 1: Successful completion rates

Algorithmic thinking skills were captured under the lens of algorithmic design and the use of logical structures that are necessary to address computational problems: sequence, selection and loop. The students used the logical structure of sequence by translating calculus assignments into computational steps in Geogebra, the loop through iterations to repeat the required steps that lead to the intended results, and selection through conditional statements to consider the particular cases for a general solution of problems.

The workbook guided the students in performing the steps in solving the given problems by predefining the steps that had to be filled in. Filling in the steps required an excellent synergy of understanding how to use the logical structures and mathematics knowledge (for example using AT to consider the special case for a general solution). The usual workflow was to fill in the steps in the workbook and then complete the task in GeoGebra as illustrated in Figures 2 and 3.

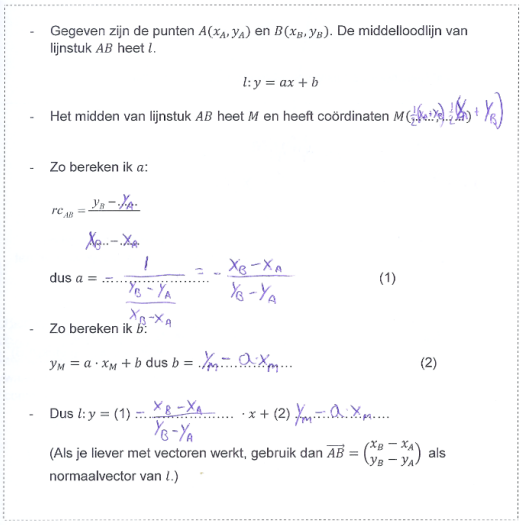

Figure 2: Task from the workbook

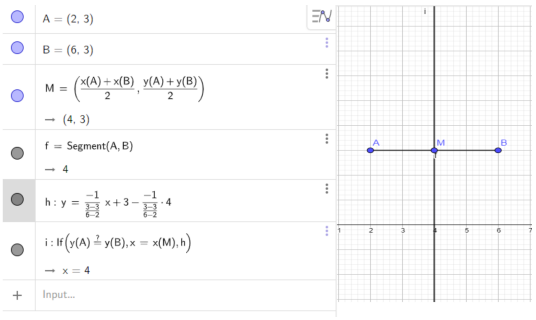

Figure 3: Example of Geogebra assignment

Generalization was a manifest part of chapters 1 to 6. In the workbooks, the tasks about the specific and general cases were generally sufficiently filled in.

4.2 Students' Experience on AT and Generalization

Students' overall reflections in the interviews indicate that they welcomed the use of computational tools in their calculus lessons. They perceived the benefits of using dynamic mathematics software as more significant than the challenges new computational tools bring. In addition, the students used AT and generalization skills to a highly satisfactory degree in the assignments (as shown in Figure 1) and elaborated on them during the interviews.

4.2.1 Students' Self-Reported Strategies for AT. Defining the steps in the workbook to solve a problem and the implement the designed solution in GeoGebra was mentioned as useful and handy in the interviews by 10 students:

> "Yeah, because ... if you write it down, it is easier to memorize. And then put it all in the computer in one go" (Student14, lesson 3)

> "It is handy, because you can fall back on it if you get lost, if you are overwhelmed by the numbers" (Student9, lesson 5).

However, one of the students had an explicit preference for first working out the task in GeoGebra and then writing down the steps in the workbook.

> "Well, I actually do it the other way round. I first fill it in (GeoGebra), and then I look if it is correct, and then I write it down. But I find it useful ... because ... you can see step by step what you do" (Student5, lesson 3).

Translating the calculus assignments into computational steps in GeoGebra, also required decomposition skills to break down the problem into smaller, more manageable parts that some students refer to as *"steps"*, as Student5 illustrated:

> "In the exercise, it is stated that you need to do it step by step and it becomes more clear to me...It just helps me very much" (Student5, lesson 3).

In the Chapter 7 end report, a student described the steps of the Newton Raphson method, expressing his understanding of using commands for the implementation of the computational steps in GeoGebra:

> "After entering the above steps, such as derivative and tangent, into GeoGebra, point B had been calculated, using the Iteration option of GeoGebra ... these steps were repeated five times, thus the zero point was accurately determined." (End report of Student10 explaining the steps to calculate zero point).

In the Chapter 7 end report, a student showed understanding of the advantage of using a macro in GeoGebra by explaining:

> "To do these steps for each point on the function is not practical. GeoGebra has a solution for this: you take all these steps together in a so-called 'macro'." (End report Student14).

4.2.2 Students' Self-Reported Strategies for Generalization. Students worked on calculus tasks starting with a specific solution and then proceeded with finding a general solution. The shift to finding the general solution seemed natural to most students. In the interviews, seven students mentioned it as easy. Student4 (in lesson 6) explained:

> "[Generalising] went well, because you started with an example, with all the intermediate steps, so you only had to replace it with letters".

Another student (Student1, lesson 4) also mentioned that it is handy and easy to use letters. On the other hand, a student (Student13, lesson 4) explained she did not find it easier, but she sees it is handy to only fill in letters. One of the five students who completed the whole series, including the more challenging and voluntary chapter 7 final report, reflected on the Newton-Raphson method's general use for calculating zeros of a function. She described the limitations of the method with regards to generalization as:

> "There are some limitations to this method, though: ... If the derivative is equal to zero. To calculate the new x-value (x1) you must divide by the derivative. If it is

> 0 the method does not work since dividing by zero is not possible" (Student12).

4.3 Teacher's View on Addressing AT and Generalization

With respect to AT and using formulas, the teacher encouraged students to let the computer do the work and stressed the advantage of using functions in GeoGebra:

> "Calculate y'(x), while GeoGebra can do it, is a pity.... Because then the function is no longer adaptable into something else".

Overall, the students showed a good understanding of the logic structures of sequence, loop and selection. The teacher was particularly satisfied with the successful completion rate of the assignments that required the use of iterations and conditional statements.

With respect to generalization, the teacher experienced that using variables in GeoGebra triggered insight about the usefulness of variables. He described in the logbook:

> "Many don't realize that if you define... 'a', then you can then call 'a'. So they just type in the formula again... For some, this is really an 'eye-opener'.".

4.4 Encountered Challenges

4.4.1 Common Mistakes in Workbooks and GeoGebra. The evaluation of the students' workbooks and GeoGebra files aimed to identify which computational concepts and practices of the designed material were challenging or problematic for the students. Therefore, this evaluation does not seek to characterize the students' learning outcomes in AT and generalization. Instead, it captures students' ways of using these skills and their potential misconceptions related to using logical structures (e.g., selections and loops) and parameters or variables.

The workbook analysis in Figure 4 shows that the tasks that were the most challenging for the students were related to a) conditional statements (11 students), iteration mistakes (e.g., wrong steps) (7 students), wrong equation being used (4 students), missing the general case (4 students) and incorrect use of variables (3 students).

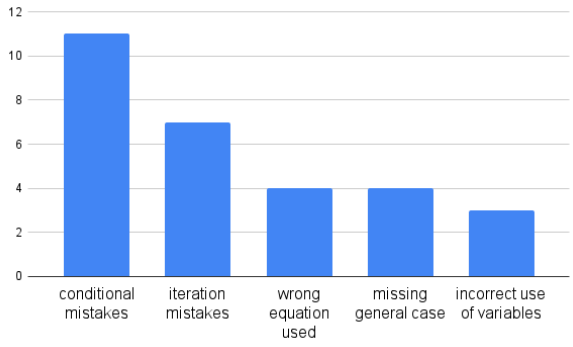

Figure 4: Workbook mistakes

The GeoGebra tasks that appeared to be the most challenging (Figure 5) for the students were related to missing part of the solution (9 students), even though it is unclear if students did not understand the assignment - 6 of the students did not include their own example in chapter 6. Seven students included parts of code that did not serve a purpose to the solution of the given problems. Other issues included the use of iterations (3 students) and conditional statements (3 students). Some students did not manage to properly save their file (3 students exported one assignment file as a picture).

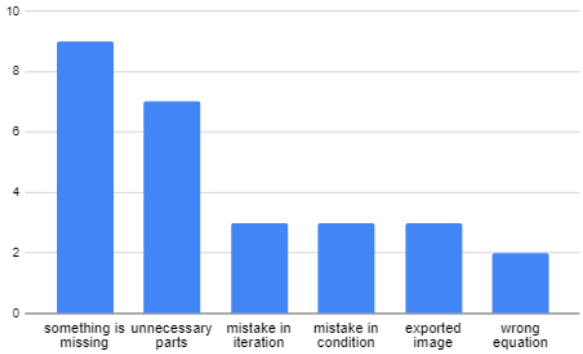

Figure 5: GeoGebra issues

4.4.2 Perceived Difficulties by the Students. The interview analysis suggests that students did not meet significant hurdles that were too challenging to address during the learning activity. Most students (10 out of 12) stated that the assignments were not particularly hard but "manageable". However, two students (Student5 and Student13) found the workbook assignments and the computational environment very challenging halfway during the lesson series. Still, in the interview in the last lesson, one of the two students (Student13) stated that she is now more familiar with using GeoGebra and the calculus tasks, and the assignment isn't that frustrating as she initially thought.

Most students (9 out of 12) stated that it took some time to get used to the GeoGebra syntax. Three students mentioned that they did not have particular problems with the syntax of the commands, and it was simple to fix typos. In addition, some of the students (7 out of 12) mentioned getting used to the GeoGebra environment (e.g., buttons, functions and navigating the GeoGebra environment) was quite difficult.

Encountered problems were related to the translation of normal language into formulas, as explained by a student:

> "I think it's more that I know ... what it should be, but I don't know how to formulate" (Student12, lesson 4).

Students especially struggle with the exact syntax. There are quotes by students related to struggles using commas and periods (Student13, lesson 6), lower- or uppercase (Student1, Student13 in, lesson 4), typing in long formulas and not knowing what is wrong (Student5, lesson 4), or unexpected things happening while executing the code (Student2, lesson 5). Occasionally, there were

unknown issues that were solved by asking the teacher or peers for help (Student9, lesson 5), or trying again (Student2, lesson 5).

Seven students mentioned problems when implementing their solution in GeoGebra in the beginning of the lesson series. Specifically, 4 students mentioned conditional reasoning is difficult. A student (Student5) who worked out the conditional statements on paper first, while she was the one who preferred to work in GeoGebra before writing down in the workbook while reflecting on the use of the if-then-else construct, stated:

> "That was something I found really difficult! That was the most difficult thing", "because ... on the one hand, you can think logically, but I find that difficult, to think if you do this what happens or what happens otherwise. I find that the most difficult".

One student got frustrated in GeoGebra:

> "I'm getting so super frustrated with this that I've almost thrown the computer 3 times already" ... "Then I type one thing wrong and then I don't see what I'm typing wrong. And then suddenly an 'area' comes up and then I do it three times and three times again an area comes up. And then something just goes wrong and I don't see what goes wrong" (Student13, lesson 4).

She characterizes herself as being not precise. Later in the lesson series, this frustration disappeared and she expressed what she learned from the lesson series:

> "...that I start thinking more precisely in terms of how I should write it down and that my notation becomes more (precise) than it was at first" (lesson 6).

Some students struggled with using variables. For example, a student (Student5, lesson 4) explicitly stated that she found it challenging to work with "letters" instead of numbers:

> "That you have to work a lot with letters and not with numbers. I find that difficult. So today, we had a + b + c, and I find that more complicated than if you don't use that. That you kind of have a lot of unknowns then".

Two of the students stated they encountered problems in properly saving the GeoGebra files. This was also reflected in the teacher's logbook and the lower number of GeoGebra files we collected (11 out of 15 students handed in their GeoGebra files).

The students mentioned different resources for tackling the problems they encountered during the learning activities. All students used workbooks that included hints or detailed explanations and could discuss with their teachers and peers. In particular, most students (7 out of 12) explicitly mentioned their teacher's support for tackling occurring issues, assistance from their peers (4 out of 12), and the use of the workbook's hints (4 out of 12).

5 DISCUSSION

In this study, we aimed to answer the following research questions:

- **RQ1**: How can AT and generalization aspects be addressed using DMS in 12th-grade calculus lessons?
- **RQ2**: What challenges do students encounter in successfully completing AT and Generalization tasks using DMS in 12th-grade calculus lessons?

With respect to the first research question about how the aspects of generalization and algorithmic thinking can be addressed in the lesson series, we found that students managed to solve problems that required generalization skills and the employment of algorithmic thinking skills in GeoGebra. Therefore, we infer that these aspects were fruitfully employed by the students during the lessons. Apparently, the design of using a workbook to define steps on paper and implementing the algorithms in the dynamic mathematics software GeoGebra successfully addressed the aspects of generalization and algorithmic thinking. This is in line with studies that advocate to teach CT both plugged and unplugged [6, 9]. The combination of plugged and unplugged work seemed a successful strategy to address aspects of CT.

The students perceived the lesson series as manageable with some challenges along the way, and eventually fun to do. This was confirmed by the teacher, who experienced that the students gradually completed the tasks with more ease and were very enthusiastic.

With respect to the second research question about the challenges students encountered, we found that students mainly had difficulties in getting to know the software and finding out the specific syntax rules. This is in accordance with previous findings related to novices using computational tools (especially in programming) and the required time to familiarize themselves with commands, e.g., [37]. An idea to overcome the difficulties in defining formulas and commands in GeoGebra and support understanding of the logic might be to offer help in structuring the formula and highlighting elements, as done in programming environments, for example by annotating functions and thereby putting focus on decomposition of the function [41]. This scaffolding might shorten the initial phase of learning how to use the technology and enable a larger focus on the content and the related CT aspects.

Another challenge students encountered was for the part of algorithmic thinking the use of logical structures with regard to formulas and commands on GeoGebra. Logical reasoning skills is a known crucial skill in general [8] and specifically in learning to program [22]. As such, this finding indicates that teachers who want to address CT might need to provide extra support or scaffolding for the learning of this aspect of CT.

The teacher was highly familiar with the materials, the tool, and the concept of CT to be taught. In general, there are less favorable circumstances and teacher training and guidance is necessary, to use ICT and CT effectively in the classroom [5]. To help teachers integrate CT in the classroom, workshops and co-designing a CT enhanced curriculum have proved to be fruitful means [48]. Moreover, teachers who are less experienced in CT might practice CT partly and miss some aspects of CT [31], and thereby provide limited opportunities to foster CT. In order to facilitate the teaching of the lesson series for teachers who are less familiar with CT, we developed a teacher guide to provide advice for the didactical approach and support for the use of the lesson series with respect to the intended aspects of CT.

We implemented the lesson series in a small-scale case study with a single class of 15 students. This limits the power of our findings. While we have an indication of the learning process and the challenges involved in the setting of our study, further research on a larger scale is required with different types of classes, different levels of experience of the teacher, and different teaching circumstances. Especially, the case of online education involves more challenges, as was experienced during the COVID-19 pandemic and which may be part of our future education [11].

Our focus was explicitly on the aspects of generalization and algorithmic thinking for the topic of calculus in dynamic mathematics software GeoGebra. The learning of functions is complex and very suitable to learn about processes that require input and produce output, and to see the relation between the varying input and accordingly changing output [35]. However, there is a broad range of topics within mathematics useful for the development of CT skills that have been addressed in research previously and need to be further explored, for example geometry and algebra [4] and statistics [18]. Our study is an in-depth exploration of addressing CT aspects within 12th-grade calculus, providing ideas for opportunities and next steps to address CT aspects in mathematics education research and practice.

6 CONCLUSION AND FUTURE WORK

In conclusion, GeoGebra seemed to be an appropriate tool to address CT aspects such as algorithmic thinking and generalization in a 270-minutes lesson series for 12th-grade students about calculus. The students and the teacher have a positive perception of the CT lesson series and the related learning process, and students fruitfully worked on the tasks, after overcoming initial challenges with the technology and getting acquainted with the CT ideas behind the design of the materials.

Future research might focus on larger scale teaching experiments with elaborated lesson series, covering a wider variety of topics within mathematics. For more insight into the learning outcomes of CT, targeted assessment and standardized pre- and post tests need to be designed to capture CT skills in more detail. Finally, teachers' professional development might enhance a better integration of CT in mathematics education.

ACKNOWLEDGMENTS

The research reported here was part of a larger project, (partly) financed by the Netherlands Initiative for Education Research, project number 00517751.

REFERENCES

[1] Folake Modupe Adelabu, Moses Makgato, and Manto Sylvia Ramaligela. 2019. Enhancing learners' geometric thinking using dynamic geometry computer software. *Journal of Technical Education and Training* 11, 1 (2019).

[2] Charoula Angeli, Joke Voogt, Andrew Fluck, Mary Webb, Margaret Cox, Joyce Malyn-Smith, and Jason Zagami. 2016. A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. , 47–57 pages. https://doi.org/10.2307/jeductechsoci.19.3.47

[3] Soumela Atmatzidou and Stavros Demetriadis. 2016. Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems* 75 (2016), 661–670. https://doi.org/10.1016/j.robot.2015.10.008

[4] Thiago S Barcelos, Roberto Muñoz-Soto, Rodolfo Villarroel, Erick Merino, and Ismar Frango Silveira. 2018. Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review. *J. UCS* 24, 7 (2018), 815–845.

[5] Kaan Bati and Mehmet İkbal Yetişir. 2021. Examination of Turkish Middle School STEM Teachers' Knowledge about Computational Thinking and Views Regarding Information and Communications Technology. *Computers in the Schools* (2021). https://doi.org/10.1080/07380569.2021.1882206

[6] Tim Bell and Michael Lodi. 2019. Constructing Computational Thinking Without Using Computers. *Constructivist Foundations* 14, 3 (2019), 342–351.

[7] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association*. Vancouver, Canada, 1. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

[8] Ruth M.J. Byrne and P. N. Johnson-Laird. 2009. 'If' and the problems of conditional reasoning. *Trends in Cognitive Sciences* 13, 7 (2009), 282–287. https://doi.org/10.1016/j.tics.2009.04.003

[9] Elisa Nadire Caeli and Aman Yadav. 2020. Unplugged Approaches to Computational Thinking: a Historical Perspective. *TechTrends* 64, 1 (2020), 29–36. https://doi.org/10.1007/s11528-019-00410-5

[10] Luis Alberto Calao, Jesús Moreno-León, Heidy Ester Correa, and Gregorio Robles. 2015. Developing mathematical thinking with scratch. In *European Conference on Technology Enhanced Learning*. Springer, 17–27.

[11] Yiming Cao, Shu Zhang, Man Ching Esther Chan, and Yueyuan Kang. 2021. Post-pandemic reflections: lessons from Chinese mathematics teachers about online mathematics instruction. *Asia Pacific Education Review* 22, 2 (2021), 157–168. https://doi.org/10.1007/s12564-021-09694-w

[12] Nancy Carter, Denise Bryant-Lukosius, Alba Dicenso, Jennifer Blythe, and Alan J. Neville. 2014. The use of triangulation in qualitative research. *Oncology Nursing Forum* 41, 5 (2014), 545–547. https://doi.org/10.1188/14.ONF.545-547

[13] Erick John Fidelis Costa, Livia Maria Rodrigues Sampaio Campos, and Dalton Dario Serey Guerrero. 2017. Computational thinking in mathematics education: A joint approach to encourage problem-solving ability. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–8.

[14] J. Cuny, L. Snyder, and Jeannette M. Wing. 2010. Demystifying computational thinking for non-computer scientists. (2010). http://www.cs.cmu.edu/$\sim$CompThink/resources/TheLinkWing.pdf

[15] E Bigotte de Almeida, A Gomes, F Correia, and R Almeida. [n.d.]. MATHSCRATCH-BRINGING PROGRAMMING AND MATHEMATICAL SKILLS INTO HIGHER EDUCATION. ([n. d.]).

[16] Paul Drijvers, Juan D. Godino, Vicenç Font, and Luc Trouche. 2012. One episode, two lenses: A reflective analysis of student learning with computer algebra from instrumental and onto-semiotic perspectives. *Educational Studies in Mathematics* 82, 1 (2012), 23–49. https://doi.org/10.1007/s10649-012-9416-8

[17] Tim Erickson, Michelle Wilkerson, William Finzer, and Frieda Reichsman. 2019. Data moves. *Technology Innovations in Statistics Education* 12, 1 (2019).

[18] Tim Erickson, Michelle Wilkerson, William Finzer, and F. Reichsman. 2019. Data moves. *Technology Innovations in Statistics Education* 12, 1 (2019). https://escholarship.org/uc/item/0mg8m7g6https://escholarship.org/uc/item/7gv0q9dc

[19] Gerald Futschek. 2006. Algorithmic thinking: The key for understanding computer science. In *Informatics Education – The Bridge between Using and Understanding Computers*, Roland. T. Mittermeir (Ed.). Springer, 159–168.

[20] Gerald Futschek. 2013. Extreme Didactic Reduction in Computational Thinking Education. In *X World Conference on Computers in Education*. Torun, Poland. http://publik.tuwien.ac.at/files/PubDat_231370.pdf

[21] George Gadanidis, Erin Clements, and Chris Yiu. 2018. Group theory, computational thinking, and young mathematicians. *Mathematical Thinking and Learning* 20, 1 (2018), 32–53.

[22] Anabela Gomes and A J Mendes. 2007. Learning to program - difficulties and solutions. In *International Conference on Engineering Education – ICEE 2007*. Coimbra, Portugal.

[23] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[24] Markus Hohenwarter, Judith Hohenwarter, Yves Kreis, and Zsolt Lavicza. 2008. Teaching and Learning Calculus with Free Dynamic Ma- thematics Software GeoGebra Calculus with GeoGebra. *Proceedings of the International Conference on the Teaching of Mathematics - TSG 16* September 2016 (2008), 1–9.

[25] Kemi Jona, Uri Wilensky, Laura Trouille, Michael Horn, Kai Orton, David Weintrop, and Elham Beheshti. 2014. Embedding Computational Thinking in Science, Technology, Engineering, and Math (CT-STEM). *Future directions in computer science education summit meeting* 2002 (2014), 1–5. http://ccl.sesp.northwestern.edu/papers/2014/OrtonKaiNorthwestern-1.pdf

[26] Maria Kallia, Sylvia Patricia van Borkulo, Paul Drijvers, Erik Barendsen, and Jos Tolboom. 2021. Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education* (2021), 1–29.

[27] Sharie Kranz, Catherine Tabor, Art Duval, Kien H Lim, Amy Elizabethwagler, and Eric A Freudenthal. 2012. iMPaCT-Math: games & activities that motivate exploration of foundational algebra concepts-while inadvertently scaffolding

computational thinking and engineered design. In *American Society for Engineering Education*. American Society for Engineering Education.

[28] Chronis Kynigos and Marianthi Grizioti. 2018. Programming approaches to computational thinking: Integrating Turtle geometry, dynamic manipulation and 3D Space. *Informatics in Education* 17, 2 (2018), 321–340.

[29] Rubin Landau, Greg Mulder, Raquell Holmes, Sofya Borinskaya, NamHwa Kang, and Cristian Bordeianu. 2014. INSTANCES: incorporating computational scientific thinking advances into education and science courses. *Concurrency and Computation: Practice and Experience* 26, 13 (2014), 2316–2328.

[30] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-smith, Linda Werner, Jalal Nouri, Lechen Zhang, Linda Mannila, and Eva Norén. 2011. Computational thinking for youth in practice. *ACM Inroads* 2, 1 (2011), 32–37. https://doi.org/10.1080/20004508.2019.1627844

[31] Swasti Maharani, Toto Nusantara, Abdur Rahman As'ari, and Abd Qohar. 2021. Exploring the computational thinking of our pre-service mathematics teachers in prepare of lesson plan. *Journal of Physics: Conference Series* 1783, 1 (feb 2021), 012101. https://doi.org/10.1088/1742-6596/1783/1/012101

[32] John Monaghan, Luc Trouche, and Jonathan M. Borwein. 2016. *Tools and mathematics : instruments for learning*.

[33] Richard Noss and Celia Hoyles. 1996. The visibility of meanings: Modelling the mathematics of banking. *International Journal of Computers for Mathematical Learning* 1 (1996), 3–31.

[34] OECD. 2018. *PISA 2022 MATHEMATICS FRAMEWORK (DRAFT) - OECD*. Retrieved July 26, 2021 from https://pisa2022-maths.oecd.org/files/PISA2022MathematicsFrameworkDraft.pdf

[35] Michael Oehrtman and Patrick W Thompson. 2008. Foundational reasoning abilities that promote coherence in students' function understanding. In *Making the Connection: Research and Teaching in Undergraduate Mathematics Education*. 27–42. https://doi.org/10.5948/UPO9780883859759.004

[36] Seymour Papert. 1980. *Mindstorms.Children, Computers, and Powerful Ideas*. http://delivery.acm.org/10.1145/1100000/1095592/cb-ms-papert.pdf?ip=131.211.43.58&id=1095592&acc=ACTIVESERVICE&key=0C390721DC3021FF.4AD871FF6AD78CEE.4D4702B0C3E38B35.4D4702B0C3E38B35&__acm__=1554196123_d16cdb415cb5dd8f8d2f777ce12019ea

[37] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for all. *Commun. ACM* 52, 11 (2009), 60–67. https://doi.org/10.1145/1592761.1592779

[38] Steven M Ross and Gary R Morrison. 2013. Experimental research methods. In *Handbook of research on educational communications and technology*. Routledge, 1007–1029.

[39] John Sanford. 2018. Introducing computational thinking through spreadsheets. In *Computational Thinking in the STEM Disciplines*. Springer, 99–124.

[40] Cynthia Selby and J. Woollard. 2013. Computational Thinking: The Developing Definition. In *ITiCSE 2013*. https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf

[41] Stefan Tubić, Miloš Cvetanović, Zaharije Radivojević, and Saša Stojanović. 2021. Annotated functional decomposition. *Computer Applications in Engineering Education* (2021), 1–13. https://doi.org/10.1002/cae.22394

[42] Jan Van den Akker, Koeno Gravemeijer, Susan McKenney, and Nienke Nieveen. 2014. *Educational design research*. 131–140 pages. https://doi.org/10.1007/978-1-4614-3185-5_11

[43] Tom Verhoeff. 2009. 20 years of IOI competition tasks. *Olympiads in Informatics* 3 (2009), 149–166.

[44] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147.

[45] David Weintrop, Daisy Wise Rutstein, Marie Bienkowski, and Steven Mcgee. 2021. Assessing computational thinking: an overview of the field. *Computer Science Education* 31, 2 (2021), 113–116. https://doi.org/10.1080/08993408.2021.1918380

[46] Uri Wilensky, Corey E. Brady, and Michael S. Horn. 2014. Fostering computational literacy in science classrooms. *Commun. ACM* 57, 8 (2014), 24–28. https://doi.org/10.1145/2633031

[47] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[48] Sally P.W. Wu, Amanda Peel, Connor Bain, Gabriella Anton, Michael Horn, and Uri Wilensky. 2020. Workshops and co-design can help teachers integrate computational thinking into their k-12 stem classes. In *Proceedings of International Conference on Computational Thinking Education*, Vol. 63. 63–68. http://tinyurl.com/netlogofire;

[49] Bedrettin Yazan. 2015. Three approaches to case study methods in education: Yin, Merriam, and Stake. *The Qualitative Report* 20, 2 (2015), 134–152. https://doi.org/10.22347/2175-2753v8i22.1038