

# From Absent Diagnosis to Deep Intervention: How Computational Thinking Shapes Mathematics Teachers' Pedagogical Guidance

Feyza Kurban

*Ministry of National Education, Burdur, Türkiye*

Received: February 2025 || Accepted: August 2026

© 2026 Mathematics Education Research Group of Australasia, Inc.

Although Programs that Simulate Mathematical Processes (PSMP) offer a computational approach to teaching mathematical concepts and integrating computational thinking (CT) into mathematics instruction, this study focuses on the development of mathematics teachers' pedagogical guidance when addressing student errors and misconceptions in PSMP-based tasks. It also examines the factors that influence this development. Twenty-eight mathematics teachers selected through criterion sampling completed written assessments involving PSMP-based tasks. Twelve participants were subsequently selected for in-depth interviews to capture a range of intervention strategies. Directed content analysis revealed a continuum progressing from an absence of diagnosis, characterised by incorrect, vague, or incomplete guidance, to diagnosis-based interventions ranging from misaligned to shallow or indirect guidance, and ultimately to deep diagnosis. This progression reflects differences in teachers' capacity to identify and address student errors and misconceptions. The findings also indicate a relationship between teachers' application of CT processes and the quality of their pedagogical guidance. Teachers who applied CT processes appropriately, including data processing and updating, algorithm visualisation, logical flow control, and testing and debugging, provided general or deep diagnosis-based interventions. In contrast, teachers who applied these processes inappropriately offered misguided, questionable, or fragmented support. These findings highlight the importance of CT competence in shaping the quality of mathematics teachers' responses to student errors and misconceptions in PSMP-based tasks.

**Keywords** • computational thinking • mathematics teacher education research • mathematics teachers' pedagogical computational thinking phases • diagnostic computational thinking • non-diagnostic computational thinking • guidance • impact of computational thinking processes on teacher guidance

## Introduction

The increasing presence of digital systems in society accentuates the significance of computational thinking (CT), which scholars argue is a fundamental skill akin to reading, writing, and arithmetic, and crucial for navigating the modern, digitised world (Barcelos et al., 2018; Büscher, 2025; Hickmott et al., 2018; Wing, 2006). Given this growing recognition of CT as a crucial competency, educational policymakers have started to incorporate these skills into formal education. This has led to recent curriculum reforms in many European Union Member States that have introduced CT's foundational concepts into primary and secondary education through a variety of approaches (European Commission, Joint Research Centre, 2022). In parallel with these reforms, researchers have turned to different teaching strategies, such as unplugged activities, contextual learning, and scaffolding, to effectively foster computational skills across diverse educational contexts (Sentance & Csizmadia, 2017). These strategies have been complemented by approaches like embedding CT across the curriculum, as an expectation for all subjects and teachers, rather than limiting it to a single domain like Informatics or alternatively integrating CT within existing subjects such as Mathematics and Technology (European Commission, Joint Research Centre, 2022).

CT and mathematics share overlapping practices, including complex problem solving, modelling, data analysis and interpretation, statistics and probability, pattern recognition and generalisation, and



the design and use of algorithms—highlighting their natural compatibility and potential to mutually support abstract problem solving (Lee et al., 2023; Rycroft-Smith & Connolly, 2019; Sneider et al., 2014). Engaging with mathematics through CT offers a structured framework and shared vocabulary for discussing mathematical ideas, strengthens understanding of variables and functions, fosters heuristic reasoning beyond formal knowledge, and uses non-numerical examples and simulations to illuminate complex relationships (Feurzeig et al., 2011). Given this synergy, CT concepts should be adapted effectively within formal education, so these complementary ways of thinking are leveraged to enhance learning outcomes. Integrating CT into mathematics can deepen understanding of real-world applications and strengthen learners' ability to analyse, interpret, and communicate complex information across contexts (Papert, 1996; Yadav et al., 2017). Ultimately, this integration supports abstract thinking, problem-solving capabilities, and innovative approaches needed to navigate an evolving digital landscape (Sung & Black, 2021; Yadav et al., 2014).

For teachers facing curriculum changes, developing an effective approach for delivering new content is of paramount relevance; however, introducing new material cannot be accomplished merely by acquiring additional subject matter knowledge—though this remains a crucial aspect (Sentance & Csizmadia, 2017; Sentance et al., 2013; Thompson & Bell, 2013; Thompson et al., 2013). Situated in Turkey, this study responds to recent curriculum and textbook developments that increasingly foreground CT-related competencies in mathematics (e.g., algorithmic thinking, flowcharts, and discipline-specific CT discourse), thereby intensifying demands on teachers' diagnostic guidance related to the application of Programs that Simulate Mathematical Processes (PSMP). Research indicates that many teachers feel unprepared to teach programming, despite expressing positive attitudes towards its integration (Hijón-Neira et al., 2017; Kravik et al., 2022; Mohottige et al., 2024). This reveals a substantial gap between acknowledging the salience of CT and possessing the practical skills required to teach these concepts effectively. Although initiatives such as "CT for All" by the International Society for Technology in Education, the Computer Science Teachers Association, and the United Kingdom's National Curriculum of Computing aim to promote CT integration, there remains considerable uncertainty about how best to support teachers in this endeavour (Ketelhut et al., 2020; Kong et al., 2023). A significant number of teachers report lacking the necessary knowledge and appropriate methodologies to integrate programming and CT concepts into their lessons (Kravik et al., 2022), suggesting a need for targeted professional development that strengthens teachers' integrated technological, pedagogical, and content knowledge (TPACK; Mishra & Koehler, 2006) in relation to CT (Saritepeci, 2022).

To effectively integrate CT into mathematics lessons, mathematics teachers must develop strong pedagogical content knowledge that combines mathematics and CT concepts (Dahshan & Galanti, 2024; Ye et al., 2023), acquire proficiency in core CT skills such as decomposition, pattern recognition, abstraction, and algorithmic thinking (Gadanidis et al., 2017), attain technological proficiency in using relevant tools such as programming environments and data analysis software (Dogan et al., 2021), and adapt and modify existing curricula to seamlessly incorporate CT activities (Aydeniz, 2018), while also developing assessment strategies to effectively evaluate students' CT learning in mathematics or vice versa (Weintrop et al., 2021). If mathematics teachers develop these skills, they will be better equipped to create richer learning experiences that highlight the synergistic relationship between CT and mathematics. This, in turn, enables mathematics teachers to design learning environments where students can meaningfully connect mathematical concepts with their practical applications in CT (Panselinas et al., 2020; Sung & Black, 2021). By introducing students to programming environments and computational problem-solving strategies, teachers can provide CT-based mathematics learning experiences that allow students to explore multiple solution paths, build resilience, and transform their perception of mathematics from a rigid set of formulas to a more dynamic and contextually relevant subject (Castle, 2023).

## Research Focus: Aim and Questions

As technology-driven innovative curricula reshape education, it is increasingly important that teachers are equipped with the pedagogical competencies needed to develop students' analytical, flexible, and innovative skills, including mathematical reasoning and CT capabilities. These aims can be supported through instructional practices that emphasise problem solving and modelling—for example, by engaging students in representing and manipulating functions, testing hypotheses, and creating simulations to explore real-world phenomena through programming. One way for teachers to reflect the synergy between mathematics and CT is to incorporate activities involving PSMP, which digitally model mathematical concepts, operations, and problem-solving strategies using CT concepts and processes such as sprites, variables, code blocks, loops, and conditions. Because diagnosing and addressing students' errors in PSMP-based tasks is important for supporting both mathematical understanding and CT development, this study aimed to describe the developmental phases of teachers' pedagogical guidance in response to students' errors or misconceptions. It also aimed to examine whether these phases varied depending on the appropriate or inappropriate use of CT processes. In this context, the following two research questions were addressed:

*RQ1: What are the developmental phases of teachers' pedagogical guidance when addressing students' errors or erroneous thought processes in tasks involving programs that simulate mathematical processes?*

*RQ2: How does the appropriate versus inappropriate use of computational thinking processes influence the progression of teachers' pedagogical guidance phases?*

## Methodology

Based on the directed qualitative content analysis (DQCA) steps outlined by Mayring (2000) and Assarroudi et al. (2018), the methodology of this study was structured as shown in Figure 1. Qualitative content analysis derives themes and categories from data, whereas DQCA uses theoretical frameworks to systematically code and analyse textual information (Assarroudi et al., 2018; Mayring, 2000; Zhang & Wildemuth, 2009). This approach was used to validate, refine, or extend theory by identifying categories, themes, and patterns based on predetermined coding schemes that may corroborate or challenge existing research (Assarroudi et al., 2018; Mayring, 2000). Sample selection, data-collection tools and procedures, and data analysis were conducted from a DQCA perspective. These decisions were informed by two theoretical frameworks: phases of pedagogical guidance in CT-integrated mathematics tasks and CT processes.

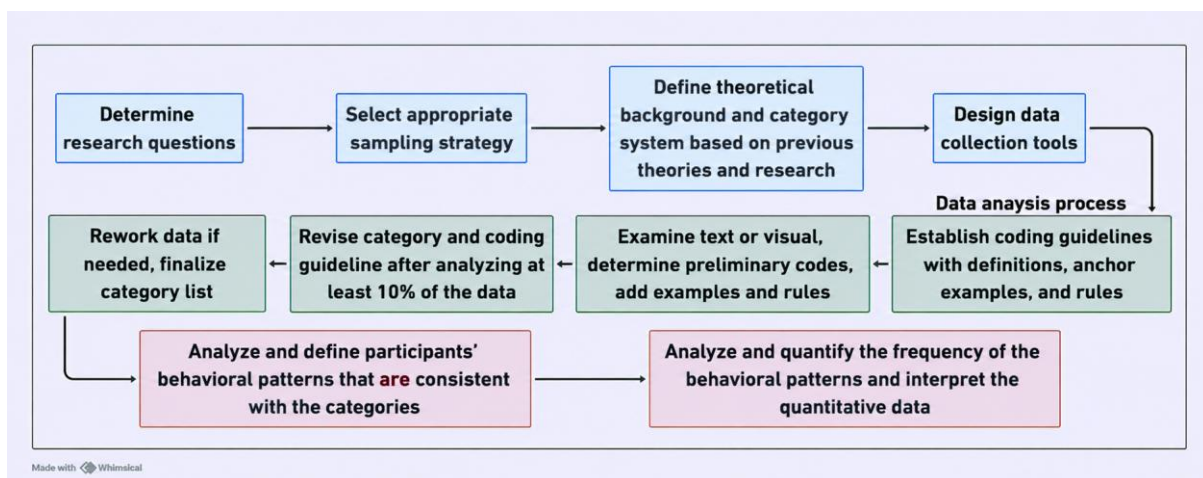


Figure 1. Research process structured through directed content analysis.

## Participants

In Assarroudi et al.'s (2018) DQCA study, purposeful sampling was recommended as a participant selection technique for obtaining specific knowledge or experience relevant to the research question, thereby enabling an in-depth investigation of the phenomenon (Palinkas et al., 2015). Within this study, that strategy was employed by way of criterion sampling, which was used to select participants with direct experience relevant to the research focus and to support an in-depth examination of factors shaping successful or unsuccessful guidance (Ahmad & Wilkins, 2025). Four criteria were considered in the selection of participants: (1) demographic characteristics; (2) technological competence; (3) willingness to integrate CT into mathematics teaching; and (4) coding or programming experience.

### *Demographic characteristics*

The sample comprised 28 mathematics teachers (10 males and 18 females) from middle and high schools in the same geographic region in Türkiye. This was intended to ensure consistency in the educational environment and minimise potential variability associated with different regional contexts. The participants' professional experience ranged from 5 to 25 years: eight teachers had 5–10 years of experience, six had 11–15 years, six had 16–20 years, and eight had 20–25 years.

From the total pool of participants—all of whom completed the PSMP-based written assessment tasks—12 were selected for interviews to capture variation in written performance. Their written work reflected a range of intervention strategies, including both developed and underdeveloped pedagogical guidance across tasks, and variation in CT process use, including both appropriate and inappropriate applications.

### *Technological competence*

Participants were teachers who reported using technology to support mathematics instruction. Most (89%) indicated frequent integration of technology and regular use of tools such as Web2 applications, GeoGebra, the Educational Informatics Network, and Mathigon, while 11% reported using these tools occasionally.

### *Willingness to integrate computational thinking into mathematics teaching*

Given the importance of teachers' willingness to integrate CT for the study, the sample intentionally included teachers who reported openness to CT integration. Specifically, 82.1% expressed a desire to integrate CT approaches into their lessons, while 7.1% indicated that their willingness was contingent upon acquiring the necessary qualifications.

### *Coding or programming experience*

All participants completed the same 32-hour course centred around Dr Scratch activities, which aimed to help teachers internalise CT processes by connecting mathematical tasks with programming, ultimately enabling them to design such programs independently.

## Theoretical Background

### *Pedagogical Guidance Phases of Teachers Related to Programs that Simulate Mathematical Processes*

The first theoretical framework, which delineates developmental pedagogical CT guidance phases, was developed by synthesising prior research on diagnosis-based versus non-diagnosis-based interventions (Seifried & Wuttke, 2010), teachers' responses to student errors in mathematical tasks (Korkmaz & Alkan, 2023), and the pedagogical implications of limited coding experience (Ketelhut et al., 2020; Mouza et al., 2017; Sentance & Csizmadia, 2017).

### *Absence of Diagnosis*

Limited experience with PSMP and CT can constrain teachers' ability to diagnose errors and students' misconceptions, resulting in three guidance phases. *Incorrect Guidance* occurs when teachers misinterpret the reasoning underlying an error in a student-designed PSMP or a student's understanding of how a PSMP operates, providing inaccurate explanations that reinforce misconceptions without addressing their underlying causes. *Vague Guidance* reflects a superficial understanding of PSMP principles, resulting in unclear or imprecise support. *Incomplete Guidance* occurs when teachers explain how to debug a PSMP by referring to its function and operational mechanisms but fail to identify the student's specific errors or misconceptions or explain why they occurred. Consequently, although such guidance may offer some assistance with debugging, it does not adequately address the underlying difficulties in students' reasoning.

### *Diagnosis-based intervention*

When teachers recognise an error in a student-designed PSMP or flawed reasoning about a PSMP, they may provide three types of guidance. *Misaligned Guidance* occurs when teachers' responses fail to resolve the error or misconception or introduce further errors or misunderstandings. *Shallow or Indirect Guidance* involves providing general or indirect support that partially addresses the issue but does not develop students' conceptual understanding of the PSMP process or the underlying CT principles. In contrast, *Deep Guidance* adopts a student-centred approach to addressing errors, misconceptions, and flawed reasoning, supporting deeper understanding while foregrounding the CT skills embedded in mathematics. This guidance may take the form of direct explanations and demonstrations, in which teachers explain why an error occurred and model a correct solution, or guided self-discovery, in which teachers prompt students to correct their PSMP or reasoning through step-by-step support.

## *Computational Thinking Processes Utilised in Programs that Simulate Mathematical Processes*

The second theoretical framework of the study is related to the CT processes utilised in PSMP. These processes were defined based on prior literature:

1. *Data Processing and Updating*: Data, often stored in variables, are processed through algorithms to perform specific actions or operations. Variable values are updated by replacing an existing value with a new one (e.g., Fagerlund et al., 2021; Weintrop et al., 2016).
2. *Algorithm Visualisation*: Abstract procedures are represented through dynamic, step-by-step visualisations that show data transformation and program execution in real time (e.g., via animations, simulations, or graphical representations) (e.g., Grivokostopoulou et al., 2017; Shaffer et al., 2010; Yamashita et al., 2016).
3. *Control of Logical Flow*: Program execution is organised as a sequence of steps in which decision points (e.g., if-else statements and logical operators such as and, or, not) determine which actions run under specific conditions (e.g., Beecher, 2017; Fagerlund et al., 2021; Grover & Pea, 2018).
4. *Pattern Recognition and Generalisation*: Identifying recurring structures, relationships, and regularities in data, code, or simulations supports abstraction and transfer, enabling the creation and improvement of PSMP and the prediction of behaviours (e.g., Barrón-Estrada et al., 2022; Beecher, 2017; Léonard et al., 2022).
5. *Decomposition and Coordination*: Complex PSMP are broken into manageable sub-systems (e.g., algorithms), and these parts are coordinated through effective connections so that the overall system functions coherently (e.g., Beecher, 2017; Fagerlund et al., 2021; Rich et al., 2019).
6. *Modelling and Optimisation*: PSMP model simplified versions of phenomena to support understanding, explanation, and prediction, often by testing hypotheses through dynamic computational environments (e.g., animations and simulations). Optimisation improves efficiency through code refinement (e.g., Beecher, 2017; Fagerlund et al., 2021).



7. *Testing and Debugging*: Testing involves systematically checking whether PSMP behave as intended, while debugging involves identifying and resolving errors in the code and in students' underlying reasoning about how PSMP work (e.g., Bowers et al., 2023; Sun et al., 2024; Vourletsis et al., 2021).

## *Data Collection Tools and Procedures*

The study's data collection tools comprised written assessments, which took about 90 minutes per participant, and task-based interviews, which lasted approximately 60 minutes each. The data collection instruments were designed to address the research questions and the two theoretical frameworks.

### *Evaluation tasks*

Ten structured written evaluation tasks were used, each involving a student PSMP containing bugs that modelled a mathematical process or performed a mathematical function, or a scenario demonstrating flawed student reasoning about the PSMP's working mechanism. During the written assessment, participants made their drawings on the assessment sheets. The tasks were aligned with the national mathematics curriculum typically addressed across Grades 7–9 (Ministry of National Education, 2018a, 2018b), reflecting the participant group of lower secondary (Grades 7–8) and early upper secondary mathematics teachers (Grade 9). Participants explained how each PSMP worked, identified the source of the bug or flawed reasoning, and described how they would guide students to correct it. The written data were analysed to determine if guidance was diagnosis-based and to identify the guidance phase, ranging from an absence of diagnosis to diagnosis-based intervention. Because PSMP tasks require different CT processes, participants' explanations were also used to identify which CT processes they applied appropriately or inappropriately. As an example of the tasks undertaken by the participants, Task 1, which involved data processing and updating, algorithm visualisation, and testing and debugging, is reproduced in Figure 2 (see next page).

### *Interviews*

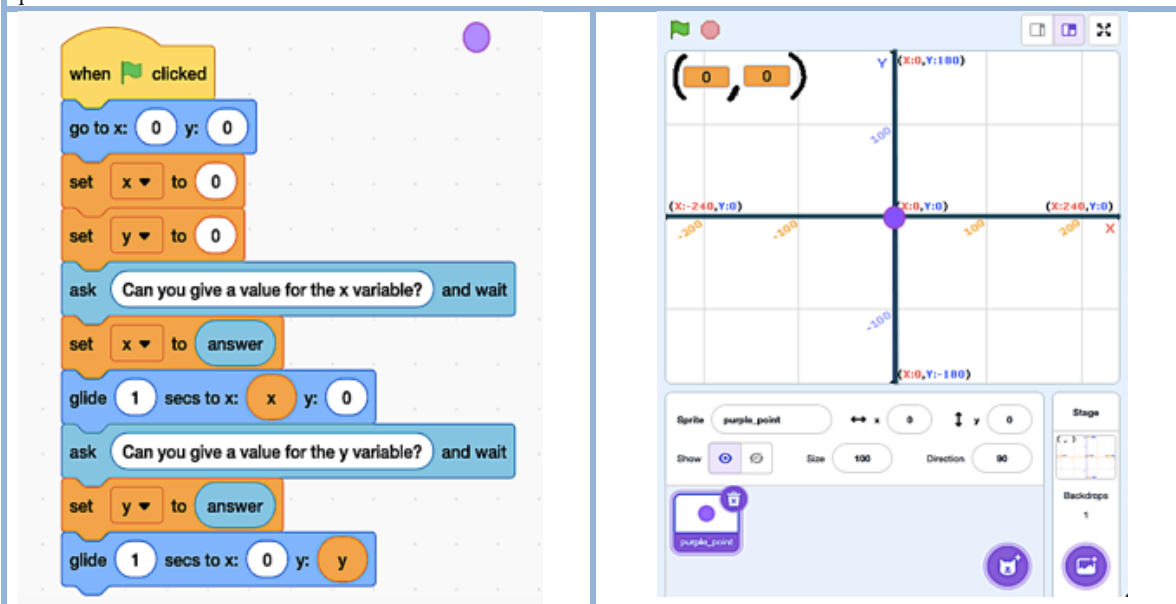
Task-based interview (Maher & Sigley, 2014) questions were designed to comprehensively reveal the participants' pedagogical guidance phases and their behavioural patterns during both successful and unsuccessful uses of CT processes. In the interviews, participants were asked the following questions to gain insight into their thought processes:

1. Could you provide a detailed explanation of how the algorithm or algorithms in the current program function?
2. Can you draw the shape that will be generated when the program runs?
3. What insights can you provide about the student's thinking process?
4. Is the PSMP designed by the student working as expected?
5. If not, where and how do you think the student made an error?
6. What do you think is the cause of the student's error or bug?
7. How can the error or bug in the PSMP design be fixed or troubleshoot?
8. How would you guide the student to correct their misunderstanding of how the PSMP works?
9. If the program reflected the student's misconceptions, how should the code blocks be arranged?

During these interviews, participants used the Scratch task materials and their own drawings to explain what they had done and their reasoning. Furthermore, a semi-structured interview strategy (Ayres, 2008) was adopted to explore further ideas expressed and to capture each participant's unique perspectives and thinking.



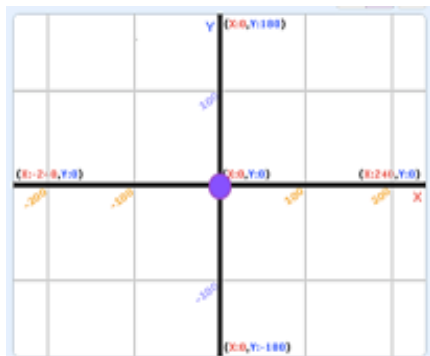
Sena aims to develop an application that displays the position of a point with user-defined  $x$  and  $y$  coordinates on a Cartesian coordinate system. Sena intends to first take inputs for  $x$  and  $y$  values from the user sequentially and then, based on these values, move a purple point, initially positioned at the origin, first along the  $x$ -axis and then parallel to the  $y$ -axis, to reach the point specified by the user ( $x$ -coordinate,  $y$ -coordinate). The code for the program Sena created for this application is provided.



Share your thoughts on whether the program designed by Sena functions as intended. Draw the possible movements of the purple point on the coordinate system when Sena's program is executed. Describe the program's working mechanism by linking it to the relevant code blocks. If you believe there is an error in Sena's design, discuss why the error occurred. Finally, explain how you would guide Sena to fix her bug so that the program functions as intended.

Does Sena's program work as she intended? Yes or No. ....

Illustrate the possible movements of the purple point on the coordinate system when Sena's current program is executed.



Describe the program's working mechanism by linking it to the relevant code blocks.

If you believe there is an error in Sena's design, discuss why the error occurred. Explain how you would guide Sena to fix her bug so that the program functions as intended.

Figure 2. Task 1 as an example.

## Data Analysis Process

To ensure accuracy and consistency, a four-step directed data analysis process was applied to both the written assessments and the interview transcripts. First, key themes and behavioural patterns across all texts and visuals were examined, and categories within each theoretical framework were established with clear scope and boundaries, supported by examples and decision rules. These category definitions, examples, and rules informed the coding guidelines and enabled systematic analysis (Assarroudi et al., 2018; Mayring, 2000). Next, two experts in mathematics education and CT coded the first task using the

guidelines, checked the fit between preliminary codes and categories, and refined category definitions and rules where needed. During analysis of the first five tasks, the guidelines were iteratively revised to reflect the expanding dataset and emerging insights; expert discussions of inconsistencies and gaps further strengthened the clarity and transparency of the guidelines, supporting the validity and reliability of the process. Finally, the guidelines were reviewed and applied to the full dataset, and final checks were conducted to confirm the categories and coding. Tables 1 and 2 illustrate how coding was conducted across the two theoretical frameworks. The exemplars in Table 1 are associated with the following task scenario:

The student-designed program does not function as intended due to the bug(s) in specific code blocks and/or flawed student reasoning about the program's working mechanism.

Table 1.

*Potential Behavioural Patterns of Participants Across Guidance Phases*

| Absence-of-diagnosis Behavioural Patterns                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Absence of Diagnosis<br/>(limited diagnosis and understanding)<br/><i>Indicators:</i> Does not diagnose the bug(s) or the student's misconception/flawed reasoning; does not understand the algorithm/program mechanism; cannot determine the intended arrangement of code blocks.</p> | <p>Potential behaviours:</p> <p><i>Incorrect:</i> Congratulates the student, stating the program works as intended.</p> <p><i>Vague:</i> Advises the student to "check/review the blocks" without identifying the source of the bug or misconception.</p> <p><i>Incomplete:</i> Points to problematic blocks or suggests debugging steps but does not diagnose the underlying misconception or explain why the bug occurs.</p>                                                                         |
| Diagnosis-based Behavioural Patterns                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>Diagnosis-based intervention<br/>(accurate diagnosis and understanding)<br/><i>Indicators:</i> Diagnoses the coding error (bug) and the student's misconception/flawed reasoning; understands the algorithm/program mechanism; knows the intended arrangement of code blocks.</p>      | <p>Potential behaviours:</p> <p><i>Misaligned:</i> Suggests changes that do not resolve the bug/misconception or introduce new errors.</p> <p><i>Shallow or Indirect:</i> Identifies blocks to change and gives brief/general explanations, without supporting conceptual understanding of the mechanism.</p> <p><i>Deep:</i> Explains why the bug occurs, links it to the student's misconception/flawed reasoning, and shows (or scaffolds) how targeted changes produce the intended behaviour.</p> |

Interview analyses enabled a detailed description and identification of behavioural patterns that indicated participants' guidance phase and their appropriate or inappropriate use of CT processes within each PSMP-based task. Analysing the written evaluation papers using the framework, Pedagogical guidance phases of teachers related to PSMP, made it possible to quantify how many participants were coded as being in the absence of diagnosis phase—particularly the incorrect, vague, and incomplete phases—and how many were in the diagnosis-based intervention phase, categorised as misaligned, shallow or indirect, or deep guidance.

Table 2.

*Participant Behavioural Pattern Examples: Appropriate vs Inappropriate Use of CT Processes*

| CT Processes Targeted by the Scratch Task Scenario                                                                | Participant Behaviours: Appropriate vs Inappropriate CT Process Use                                                                                                                                                                                                                                                                                                                              |
|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Data processing and updating</i><br>(e.g., set x to 10; set y to 20; set x to x + y; set y to x + y)           | <i>Appropriate:</i> Determines updated final values correctly (e.g., $x = 30$ , $y = 50$ ).<br><i>Inappropriate:</i> Cannot determine updated final value(s) of one or more variables.                                                                                                                                                                                                           |
| <i>Algorithm visualisation</i><br>(e.g., go to (0,0); point in direction 90; move 100)                            | <i>Appropriate:</i> Mentally traces motion correctly (e.g., left-to-right along x-axis from origin).<br><i>Inappropriate:</i> Misrepresents direction/trajectory (e.g., describes movement along y-axis).                                                                                                                                                                                        |
| <i>Control of logical flow</i><br>(e.g., if (x<0 and y>50) then ... else ...)                                     | <i>Appropriate:</i> Correctly identifies which branch executes given the conditions (e.g., recognises <i>else</i> runs when conditions are not satisfied).<br><i>Inappropriate:</i> Says neither the <i>if</i> nor the <i>else</i> will run or chooses the wrong branch.                                                                                                                         |
| <i>Pattern recognition and generalisation</i><br>(e.g., repeat (8); move x steps; turn CW 60; change x by 10)     | <i>Appropriate:</i> Correctly anticipates the emerging pattern/shape from repeated steps (e.g., recognises a spiral-like hexagonal pattern).<br><i>Inappropriate:</i> Cannot mentally represent the resulting shape or describes an incorrect pattern.                                                                                                                                           |
| <i>Decomposition and coordination</i><br>(e.g., two sprites with broadcast/receive message)                       | <i>Appropriate:</i> Decomposes the scenario into components and explains coordination between event blocks (e.g., sequencing via broadcast/receive).<br><i>Inappropriate:</i> Does not recognise decomposition or the coordination mechanism between event blocks.                                                                                                                               |
| <i>Modelling and optimisation</i><br>(e.g., a bouncing-ball simulation with decreasing height)                    | <i>Appropriate:</i> Explains that the code models a ball rising and bouncing with progressively smaller jumps and optimises the long sequence by introducing a loop and a changing height variable (e.g., set $n = 100$ ; repeat: change y by n; change y by -n; set $n = 0.8*n$ ).<br><i>Inappropriate:</i> Does not recognise the modelled process and/or cannot suggest a valid optimisation. |
| <i>Testing and debugging</i><br>(e.g., the sprite reaches the wrong location due to incorrect direction/sequence) | <i>Appropriate:</i> Identifies the mismatch between intended and actual output and proposes a specific fix that makes the output match the target.<br><i>Inappropriate:</i> Claims the program works as intended or proposes changes that do not correct the fault.                                                                                                                              |

In addition, a categorisation matrix was developed to examine each participant's appropriate and inappropriate use of CT processes across tasks (Table 3). In qualitative research, a categorisation matrix is a structured tool for systematically organising and coding data, particularly in directed content analysis (Assarroudi et al., 2018). Here, the matrix specified which processes—data processing and updating, algorithm visualisation, control of logical flow, pattern recognition and generalisation, decomposition and coordination, modelling and optimisation, and testing and debugging—should be predominantly used to complete each task. The matrix was derived from the theoretical framework, grounded in prior theories, research, and literature, and served as a guideline for classifying qualitative data into predefined categories. A separate matrix was completed for each participant, after which the number of participants who used CT processes appropriately or inappropriately in each task was determined. The quantitative values derived from coding the written evaluation papers within the two theoretical frameworks were used to provide a descriptive summary of category use and to complement the qualitative analysis.



Table 3  
The Categorisation Matrix for PSMP-based Tasks

|                                        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----------------------------------------|---|---|---|---|---|---|---|---|---|----|
| Data processing and updating           | X | X |   | X | X | X | X | X | X | X  |
| Algorithm visualisation                | X |   |   | X |   |   |   | X |   | X  |
| Control of logical flow                |   | X | X |   | X |   | X |   | X |    |
| Pattern recognition and generalisation |   |   |   |   | X | X | X | X | X |    |
| Decomposition and coordination         |   |   |   | X | X |   |   |   | X | X  |
| Modelling and optimisation             |   |   | X | X |   |   |   |   | X |    |
| Testing and debugging                  | X | X | X | X | X | X | X | X | X | X  |

## Findings

The data obtained from written responses and interviews indicated that participants exhibited behaviour patterns in CT tasks involving PSMP, demonstrating that they were in the Incorrect, Vague, and Incomplete Guidance phases, stemming from Absence of Diagnosis, and the Misaligned, Shallow or Indirect, and Deep Guidance phases, grounded in Diagnosis-based intervention. Below are findings from drawings, the written responses, and interview excerpts that define these six phases in Task 1, followed by evidence reflecting the number of participants in each guidance phase across 10 different PSMP-based tasks. Behavioural patterns, drawn from two representative tasks (one concerning a student's error in the PSMP design and the other addressing the student's erroneous thinking process about how a given PSMP works), indicated that participants who were unable to appropriately utilise CT concepts and processes tended to engage in non-diagnostic guidance, whereas those who applied these concepts appropriately predominantly provided diagnostic guidance. Task-level counts of appropriate and inappropriate CT process use were employed to provide a descriptive summary that complemented the qualitative findings.

### *Participants' Pedagogical Computational Thinking Guidance Phases in Programs that Simulate Mathematical Processes-based Tasks*

Task 1's program simulates point movement in a Cartesian plane from the origin to a target  $(x, y)$  coordinate using a two-step process: a horizontal translation along the x-axis followed by a vertical translation parallel to the y-axis. The flawed program processed user responses from interaction blocks as numerical inputs for the variables  $x$  and  $y$  (data processing and updating), correctly guiding the purple dot sprite along the x-axis to reach  $(x, 0)$  from the origin but then erroneously directing it to move diagonally to  $(0, y)$  within one second, deviating from the intended algorithmic visualisation. To fix the program, the final command needed to be changed from "go to  $(0, y)$  in 1 second" to "go to  $(x, y)$  in 1 second," ensuring that the sprite moved parallel to the y-axis from  $(x, 0)$  to  $(x, y)$  according to the assigned variable values, thus demonstrating proper testing and debugging.

As illustrated in Figure 3, the trajectories described by P1, P2, and P3 did not accurately represent the movement generated by the given program. Their explanations and proposed responses to the student's error were coded as non-diagnostic guidance. Failing to identify the bug, P1 claimed that the program functioned correctly and asserted that, with the example inputs 120 and 80, the purple dot would move as intended—first from the origin to  $(120, 0)$  and then to  $(120, 80)$ —and concluded that there were no bugs in the code, thereby providing misleading (incorrect) guidance. P2 misdiagnosed the program's behaviour by claiming that the sprite would remain stationary at the origin and suggested that the  $x$  value entered by the user should be written into the final code block. This guidance was coded as *vague* because it proposed an input-dependent patch (hard-coding a single-run value) rather than diagnosing the bug mechanism in the final block or explaining how the program should preserve the  $x$ -coordinate to reach  $(x, y)$ . P3's description of the sprite's movement included an additional step that was



not produced by the algorithm. Specifically, P3 described the trajectory as a three-step path— $(0, 0) \rightarrow (x, 0) \rightarrow (0, 0) \rightarrow (0, y)$ —and depicted this in a drawing, even though the program did not contain any block that returned the sprite to the origin after reaching  $(x, 0)$ . Although P3 suggested a change that would remove the bug, the guidance was coded as incomplete because it was not grounded in an accurate trace of the program: P3 introduced a return to  $(0, 0)$  that is not produced by the code and did not explain how the final block determines the sprite's coordinates.

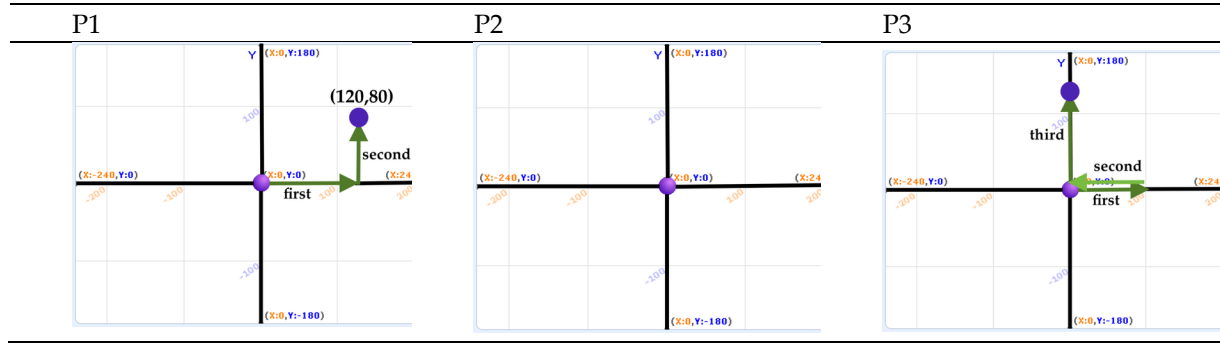


Figure 3. Behavioural patterns in the absence of diagnosis (Task 1).

P4, P5, and P6 accurately diagnosed the bug and noted that, when Sena's current program was executed, the purple point moved sequentially to  $(0, 0)$ ,  $(x, 0)$ , and  $(0, y)$  on the coordinate system. P5's written explanation exemplified this trajectory: "The point moves horizontally from the origin to a position on the  $x$ -axis determined by the input value. Then, when the  $y$  value is entered, it moves and settles at a position on the  $y$ -axis at  $(0, y)$ ." Consistent with these explanations, all three participants precisely depicted the unintended trajectory produced by the bugged program (Figure 4).

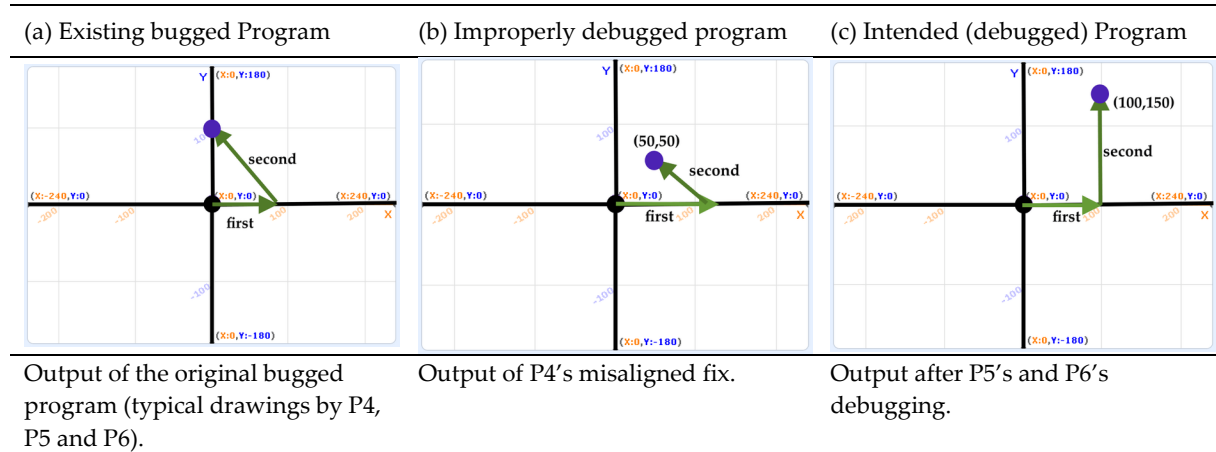


Figure 4. Program execution visuals: Bugged, Improperly debugged, and Intended.

After explaining the program's working mechanism and recognising that it did not function as intended, P4, P5, and P6 differed in the pedagogical guidance they provided to support debugging (Figure 5a–c). P4 (Figure 5a) recommended two changes: replacing  $x:(x)$  with  $x:(\text{answer})$  in the 7th block and modifying the final block from  $x:(0) y:(y)$  to  $x:(\text{answer}) y:(\text{answer})$ . This guidance was coded as *misaligned* because it relied on the most recent user input stored in *answer* (i.e., the value entered for  $y$ ), so the point reached  $(x, y)$  only when  $x = y$ ; otherwise, it ended at  $(y, y)$  (e.g.,  $x = 120, y = 50$  resulted in  $(50, 50)$ ). P5 (Figure 5b) suggested a stepwise fix: after taking the  $x$  input, setting *answer* to  $x$  and moving to  $x:(\text{answer}) y:(0)$ , then prompting for  $y$ , and finally moving to  $x:(x) y:(\text{answer})$ . Although this produced the intended endpoint, it was coded as *Shallow* or *Indirect* because the guidance did not make

the underlying storage mechanism explicit: P5 showed awareness that the  $x$ -value must be retained (by carrying it through the  $x$  variable) but did not similarly emphasise storing the  $y$ -input in the  $y$  variable and then using  $y:(y)$  in the final movement. Instead, the  $y$ -value was treated as a transient second answer, which weakened the conceptual explanation of how both coordinates should be preserved through stable variable assignments across steps. In contrast, P6 (Figure 5c) provided deep guidance by explicitly linking the error to the final movement command and the loss of variable persistence across steps. P6 therefore recommended correcting the final command to glide (1) secs to  $x:(x) y:(y)$ , explaining that the program's correctness depends on preserving both inputs in their respective variables— $x$  must remain stored in the  $x$  variable while  $y$  is stored in  $y$ —so that the final movement uses the maintained pair  $(x, y)$  rather than overwriting one coordinate and breaking the intended two-step translation.

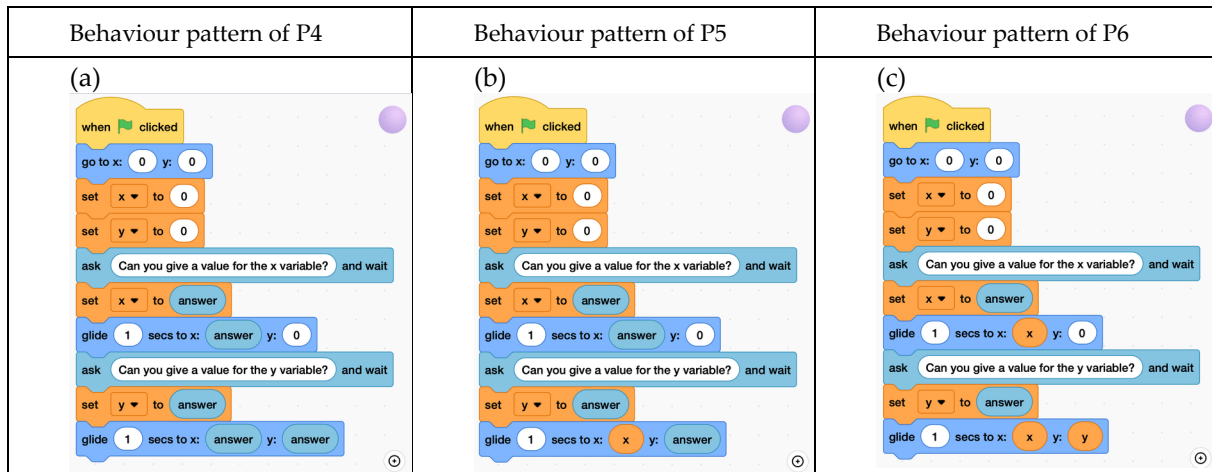


Figure 5. Behavioural patterns at diagnosis-based intervention phases (Task 1).

### Guidance Phases Across PSMP-based Tasks: From Absence of Diagnosis to Diagnosis

A spider chart displays values on radial axes extending from a common centre, with each axis representing one guidance phase. The spider charts in Figure 6 use the participant counts in the six guidance phases for each task to compare the distributions across the 10 tasks. The Average Spider Chart shows a near-even split between diagnosis-based guidance (52.5%) and absence-of-diagnosis guidance (47.5%), indicating a slight tilt towards diagnosis-based guidance (Figure 6). Tasks 1 and 6 show a perfectly balanced distribution. Absence-of-diagnosis guidance is more prevalent in Task 2 (53.6%), Task 3 (64.3%), Task 7 (67.9%), and Task 10 (57.1%); diagnosis-based guidance is more prevalent in the remaining tasks, especially in Task 4 and Task 5.

Shallow or Indirect (22.5%) and Incorrect (20.7%) show the highest overall proportions, followed by Deep (16.1%), which is close to the equal-share benchmark per phase (16.67%). In contrast, Vague (13.2%), Misaligned (13.9%), and Incomplete (13.6%) show the lowest proportions in the Average Spider Chart. The Lowest vs Highest Values Across All Tasks Chart indicates substantial task-level variation in phase proportions. Analysing the task-specific spider charts shows that Deep peaks at 39.3% in Task 5 and falls to 3.6% in Task 10. Misaligned peaks in Task 4, while Incomplete reaches its highest point in Task 7; however, Misaligned is not observed in Task 2, and Incomplete is not observed in Task 4 and Task 5. Shallow or Indirect peaks at 35.7% in Task 2 and reaches its lowest value (7.1%) in Task 7. Overall, even the least frequent phase, Vague, does not differ much from the equal-share benchmark (16.67%), and the presence of each phase across most tasks supports their overall occurrence. Despite task-level variation, the proportion of task responses involving diagnosis-based guidance—by identifying a program error or a student's difficulty in understanding a given program—closely matches the overall diagnosis-based rate (52.5%).

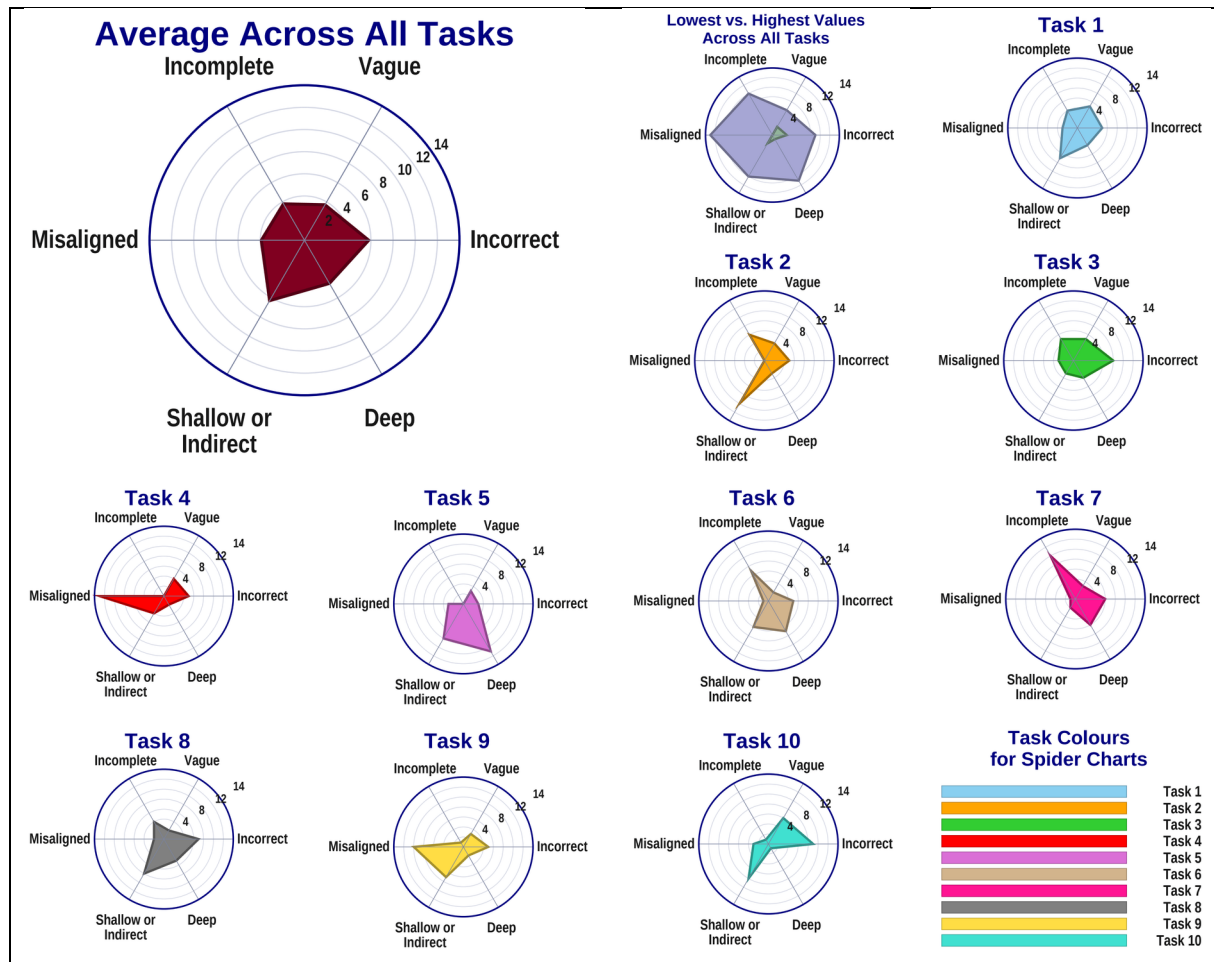


Figure 6. Spider charts depicting participant counts at different guidance phases for each task and their averages across the 10 tasks.

### *Appropriate and Inappropriate Use of Computational Thinking Processes*

In Task 3, the objective is to design a program that checks if a randomly selected number between 0 (zero) and 90 is divisible by a divisor without a remainder, displaying a message indicating the result (divisible or not) for a set duration using a “looks” block. The program begins by assigning numerical values to the divisor, dividend, quotient, and remainder variables through an algorithm triggered by clicking the green flag. Then, another algorithm is executed by clicking the Aby sprite, where it checks whether the remainder is greater than 0 (zero) and less than the divisor. This sequence of separate algorithms demonstrates decomposition, while their ordered execution showcases coordination.

Placing the quotient variable, the value obtained by the floor division of  $y/x$ , and the remainder variable, the result of the operation  $y - (\text{divisor} * \text{quotient})$ , inside the “if” and “else” sections of the conditional block, instead of under the green flag block, resulted in a bug that required testing and debugging. This bug disrupts the logical flow, as the value of the remainder variable passed into the conditional block is 0. Since 0 is not greater than 0, the number assigned as the dividend will always display “ $y$  is divisible by  $x$ ” on the screen, regardless of whether it is divisible by 5 or not.

To make the program that models the division operation functional and optimise it, the blocks that set the quotient and remainder variables should be placed immediately after the dividend variable within the event block (when the green flag is clicked), and these blocks that set the quotient and remainder variables should be removed from the conditional block (Table 4).



Table 4.

Comparative Examples of Participant Behavioural Patterns: Inappropriate vs Appropriate CT Use (Task 3)

| Inappropriate Use of CT Processes in the Absence of Diagnosis Phase |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Decomposition and coordination                                      | P2 attributed the problem to a click-trigger issue and therefore overlooked the required coordination between the two event blocks. (P2: <i>"the result might not have been displayed... because the sprite (Aby) wasn't clicked ..."</i> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Control of Logical Flow                                             | P8 noted the remainder was "0 (zero)" but failed to imply that this triggers the else branch, claiming neither branch would execute. (P8: <i>if the remainder were zero ... It doesn't go to 'if'. It also doesn't go to 'else'.</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Testing and debugging                                               | P11 assumed the program would behave as expected and therefore did not find the bug (P11: <i>I think the program is correct. I believe it will work properly.</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Appropriate Use of CT Processes in the Diagnosis-based Intervention |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Decomposition and coordination                                      | P10 explained that the program should be decomposed into two coordinated event blocks: the green-flag script assigns values (including quotient and remainder), and the Aby-click script uses these values to decide which message to display. (P10: <i>"Here, x was set to 5, and y was selected as 73 ... The remainder is already 0 here (first event block). Since the remainder wasn't assigned ... no need for it to enter such a conditional block (second event block—'when this sprite is clicked') ..."</i> ).                                                                                                                                                                                                         |
| Control of Logical Flow                                             | P6 noted that because the remainder is 0, the condition "remainder > 0" is false, so the else branch executes and "y is divisible by x" is displayed. (P6: <i>"0 is not greater than 0... it goes to the 'else'..."</i> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Testing and debugging                                               | P9 identified that the quotient and remainder blocks are wrongly placed inside the if–else structure and suggested moving them under the green-flag script, so the click-triggered script only checks the condition and displays the appropriate message. (P9: <i>"remove these two from the 'if-else'... place them ... under the green flag ... then check ..."</i> )<br>P5 implied how the blocks should be arranged to debug the program, noting that the quotient must be computed before the remainder because the remainder depends on it. (P5: <i>"It will actually calculate the quotient first because the remainder cannot be determined without the quotient."</i> )                                                 |
| Modelling and Optimisation                                          | P9 identified redundancy in computing quotient and remainder inside both branches and suggested computing them once under the green-flag script to simplify the program. (P9: <i>"These ... should not be here. One of them being under the green flag would be sufficient."</i> )<br>P5 reframed the task as modelling the division process by defining quotient and remainder through formulas (rather than assigning default values) and proposed a streamlined structure that reduces unnecessary blocks while preserving the program's logic. (P5: <i>"Let's not set the remainder and quotient to 0... make the quotient... the floor of the division, Y/X ... I used 'Y minus quotient times X' for the remainder."</i> ) |

Task 5 (Figure 7), completed by Derfu (a hypothetical student), focuses on determining the values of  $x$  and  $y$  after the 20th iteration of a nested-loop program in which both variables start at 0, with an outer loop running five times and an inner loop running four times. Within each iteration,  $y$  is updated to  $2x + y$ , followed by updating  $x$  to  $x + 5$ . Tracing these updates yields a predictable pattern: after 20 iterations,  $x = 100$  and  $y = 1900$ . The task also targets Derfu's reasoning, as the student concluded  $x = 100$  and  $y = 200$ ; participants were therefore asked to explain the program's mechanism and to describe how they would guide Derfu to recognise and correct the misconception.



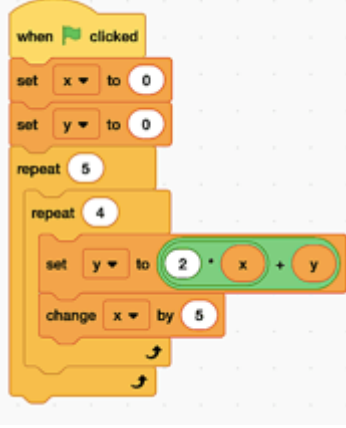
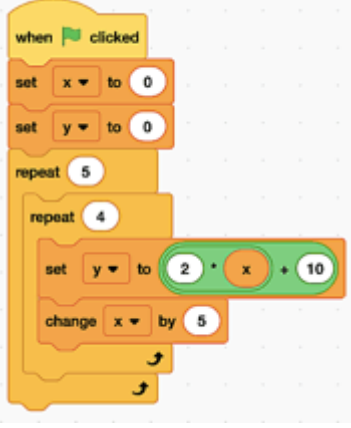
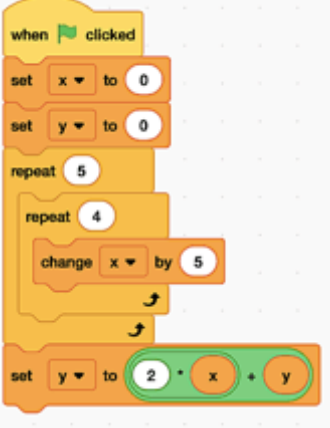
| (a) The existing program                                                          | (b) Participant-suggested program                                                 | (c) Participant-suggested program                                                   |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|  |  |  |
| The Program in Task 5 completed by Derfu                                          | The program created by P12 to represent the student's erroneous thinking process  | The program created by P5 to represent the student's erroneous thinking process     |

Figure 7. Existing program vs participant-suggested programs aligned with students' erroneous thinking (Task 5).

Table 5 differentiates participants by their data processing and updating, pattern recognition and generalisation, and their use of testing and debugging to locate the source of the mismatch between Derfu's conclusion and the program's iterative output. In the absence-of-diagnosis phase, P7 misapplied the update rules by treating  $x$  as a simple count and assuming that  $y$  resets each cycle, while P8 updated  $x$  correctly but did not treat  $y$  as cumulative when  $y$  is updated to twice the current  $x$  plus the current  $y$ . P9 relied on partial step-by-step computation and did not generalise the emerging pattern, and P10 endorsed  $y = 200$  through a shortcut rather than testing how  $y$  evolves across loop iterations. In the diagnosis-based intervention phase, P6 tracked both variables through successive cycles via systematic updating, and P11 generalised the pattern by formulating an explicit rule/series, yielding  $x = 100$  and  $y = 1900$ . Accordingly, P6 proposed checking the first few cycles to surface accumulation, P5 attributed Derfu's error to a misreading of sequential flow within the loop, and P12 suggested a code variant designed to reproduce  $y = 200$  to make the misconception explicit (Table 5).

Table 5.  
Comparative Examples of Participant Behavioural Patterns: Inappropriate vs Appropriate CT Use (Task 5)

| Inappropriate Use of CT Processes in the Absence of Diagnosis Phase |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data processing and updating                                        | P7 misread the variable-updating logic and assumed that the $y$ value did not carry over between iterations, leading to the claim that $y$ remained 0. (P7: "... $y$ always results in zero.")                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                                                                     | P8 correctly tracked the increase of $x$ but did not recognise that $y$ is cumulative, because each update depends on the current $x$ and the prior value of $y$ . This led P8 to underestimate $y$ and treat it as remaining relatively small rather than becoming much larger through accumulation. (P8: <i>I would explain to him that before <math>x</math> reaches 100, it would first be 95, and as a result, <math>y</math> would remain at 190 instead of reaching 200.</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Pattern Recognition and Generalisation                              | P9 relied on step-by-step computation of successive updates (rather than generalising a pattern) and therefore did not reach the later iterations needed for the final values. (P9: <i>"We start with <math>x = 0</math> and <math>y = 0</math>... <math>x</math> becomes 5... then 10... ... I have reached the 10th and 11th iterations."</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Testing and debugging                                               | P10 reduced the loop-based mechanism to a shortcut calculation and therefore <i>did not test</i> how $y$ accumulates across iterations; on this basis, they incorrectly validated Derfu's outcome as correct. (P10: <i>"<math>5 \times 4 = 20</math> ... <math>20 \times 5 = 100</math> ... for <math>y</math> ... <math>10 \times 4 = 40</math> ... <math>40 \times 5 = 200</math> ... Derfu didn't make any mistakes."</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                                                                     | P7 proposed a code edit, but it neither surfaced Derfu's reasoning nor reproduced Derfu's stated $y = 200$ ; instead, it generated a different (yet still incorrect) value (P7: <i>If Derfu changes <math>y</math> to <math>2 \times x + x</math> in the second-to-last code, she can find <math>y</math> by increasing <math>x</math> by 5:</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Appropriate Use of CT Processes in the Diagnosis-based Intervention |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Data processing and updating                                        | P6 approached the task through iterative updating, tracking how both variables changed at each iteration and extending this stepwise computation through to the 20th update, which reflects an appropriate use of data processing and updating. (P6: <i>"I moved forward step by step, writing each one until the 20th step, up to the 20th iteration"; e.g., <math>x = 0, y = 0</math>; <math>x = 5, y = 10</math>; <math>x = 10, y = 30</math>; ...; <math>x = 100, y = 1900</math>.</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Pattern Recognition and Generalisation                              | P11 identified that $y$ accumulates as 0, 10, 30, 60, ... and that the successive increments increase linearly, enabling a generalisation using the arithmetic-series sum. Accordingly, at the 20th iteration, $x = 100$ and $y = 1900$ . (P11: <i>"<math>y</math> takes the values 0, 10, 30, 60, ... and so on. The increase between these values forms a pattern ... From this, <math>x = 100</math>, and <math>y</math> becomes ... <math>y = 10(0 + 1 + \dots + 19) = 1900</math>."</i> )                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Testing and debugging                                               | P12 proposed a code edit intended to make the program yield Derfu's stated outcome ( $x = 100, y = 200$ ); P6 stated that they would test the first few iterations step by step to show how $(2x+y)$ accumulated and thereby surface Derfu's misconception; and P5 noted that they would correct Derfu's belief that a block operates "outside" the loop by emphasising the program's sequential flow (P12: <i>If 10 is placed instead of the <math>y</math> variable, <math>x = 100</math> and <math>y = 200</math> will be reached in the final step; P6: <i>When I explained to Derfu the values in the first three steps of the expression and the pattern, she could realise that her thinking was incorrect; P5: <i>I would explain that the program proceeds sequentially, where the output obtained from one step is used in the subsequent step. The correct answer is <math>x = 100, y = 1900</math>.</i></i></i> |

### Frequencies in the Heat Maps of Appropriate and Inappropriate Uses of Computational Thinking Processes

Across the ten tasks, participants in the Absence of Diagnosis phase struggled to apply CT processes effectively. Per task, 6–15 participants proposed ineffective approaches to identify or resolve PSMP errors, consistently struggling with testing and debugging (Figure 8). Tasks 1, 8, and 10 showed algorithm-visualisation errors among 11–13 participants. Task 3 showed decomposition/coordination challenges among 10 participants; Task 2 showed misapplied logical-flow control among 8; and Task 6



showed difficulties with pattern recognition and generalisation among 6. Figure 8 showed no recorded instances of inappropriate use of Modelling and Optimisation.

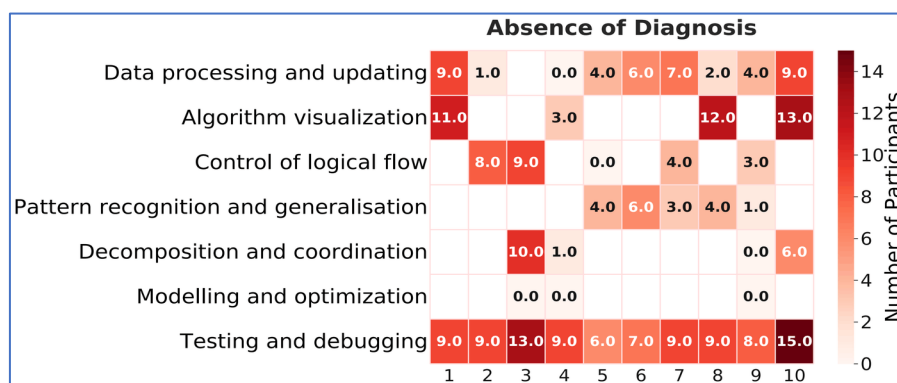


Figure 8. In the absence of diagnosis: Inappropriate use of CT processes.

In the Diagnosis-based Intervention phase, a notable number of participants (6–19) effectively employed testing and debugging across all ten tasks (Figure 9). Although Task 8 elicited the lowest engagement in data processing and updating, this process was also applied efficiently by substantial groups in other tasks (8–22 participants). Algorithm visualisation was successfully performed by 5 participants in Task 10 and by 13, 16, and 15 participants in Tasks 1, 4, and 8, respectively; in each case, participants' drawings accurately reflected the algorithm's output. A strong understanding of logical flow in PSMP, or the ability to identify related issues, was demonstrated by 13 participants in Task 2 and 22 in Task 5, and similar competence was evident in Tasks 3, 7, and 9, where 7 participants in each task managed the flow of if–else control structures effectively. Participants also demonstrated successful use of pattern recognition and generalisation, as well as decomposition and coordination, across multiple tasks. By contrast, engagement in modelling and optimisation was lower, though some participants performed successfully, likely reflecting the comparatively higher cognitive demands of this CT process.

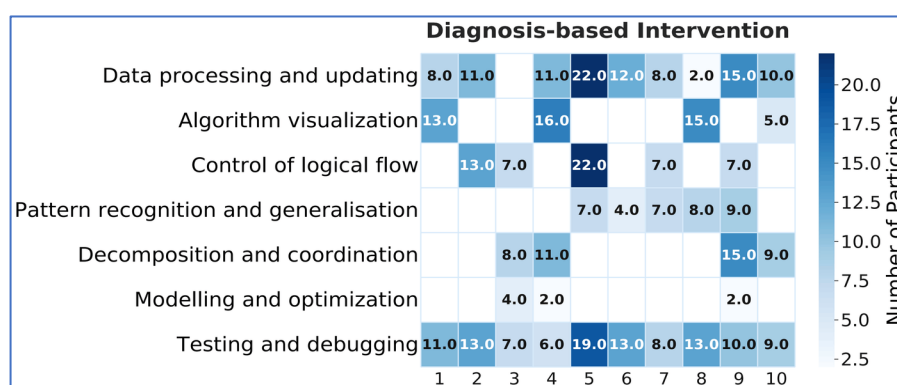


Figure 9. In diagnosis-based guidance: Appropriate use of CT processes.

## Discussion

Teachers' integration of core CT processes—decomposing complex problems into manageable sub-problems, using stepwise solution strategies, and transferring learned approaches to analogous problems—across mathematical domains depends not only on understanding CT concepts and



principles but also on the ability to convey these processes so they cohere with the mathematical content students are learning (Yadav et al., 2016; Yadav et al., 2017). Such coherence is fostered when teachers deliberately connect mathematical ideas with CT-oriented pedagogical strategies (Mouza et al., 2017). Despite not yet implementing programming or CT in their classroom teaching, participants had completed a 32-hour PD designed to internalise CT processes through PSMP-based tasks; nevertheless, the findings demonstrate that, when presented with bugs in student-designed PSMP or with errors in students' reasoning about a specific PSMP, a meaningful proportion could diagnose the error and provide effective guidance. Difficulties in diagnosis and guidance clustered among participants who misapplied the relevant CT processes; accordingly, professional development and teacher education should strengthen teachers' enactment of core CT processes in PSMP tasks—especially tracing execution and testing/debugging—and explicitly develop error diagnosis and responsive questioning to enable deep, student-centred interventions.

To prevent erroneous thought patterns from becoming habitual, teachers need pedagogical competence to diagnose student errors (e.g., logical deficiencies and faulty assumptions), select strategies to address them, and provide feedback that supports error correction (Seifried & Wuttke, 2010). The present study delineates six developmental phases of mathematics teachers' guidance in PSMP tasks—three non-diagnostic and three diagnostic—and indicates that some guidance forms do not reliably remediate students' erroneous PSMP designs or flawed reasoning about a specific PSMP. Participants' guidance also showed overlap between non-diagnostic and diagnostic forms, which is consistent with Wuttke and Seifried's (2013) view that error identification and correction remain limited until sufficient competence develops and become more sophisticated as expertise grows. In this sense, a CT education approach that foregrounds students' strengths and weaknesses can optimise learning and inform curricular and pedagogical decisions responsive to diverse cognitive needs (Garcia, 2021, 2024).

Seifried and Wuttke's (2010) distinction between diagnosis-based and non-diagnosis-based guidance, together with Korkmaz and Alkan's (2023) domain-specific analysis of teachers' responses to student errors, converges on a common structure: guidance varies by (i) whether teachers accurately diagnose the error and (ii) how far their responses function as corrective support rather than leaving misconceptions intact. Seifried and Wuttke (2010) operationalised this structure through four guidance types, combining diagnosis (present/absent) with feedback quality (limited vs detailed and correction-oriented). Korkmaz and Alkan (2023) articulated the same logic via six cases by distinguishing three non-diagnostic responses—accepting the error as correct or offering incorrect alternatives; offering no corrective explanation; or merely presenting the correct solution—from three diagnostic responses—offering no guidance; providing an incorrect solution; or explaining the error while providing the correct answer.

Applying this framework to PSMP tasks, the present study delineates six developmental phases of teacher guidance, grouped into three non-diagnostic and three diagnosis-based forms. When diagnosis is absent, guidance tends to be incorrect (misinterpreting error causes and reinforcing misconceptions), vague (superficial assistance reflecting an unclear grasp of PSMP principles and yielding imprecise support), or incomplete (providing instructions—or even the correct solution—without addressing underlying errors and misconceptions). When guidance is diagnosis-based, it is misaligned (diagnosing yet misconstruing error origins, with feedback that sustains misconceptions), shallow or indirect (general support that does not build a clear understanding of PSMP processes and CT principles), or deep (directly addressing errors via explicit reasoning with correct solutions or through guided self-discovery), the latter fostering the CT skills required for designing effective PSMP. The observed concentration in non-diagnostic and early diagnosis-based forms is consistent with constraints reported in teacher education and professional development contexts with limited coding experience, where teachers often show underdeveloped CT competencies, a superficial grasp of CT concepts, difficulty with program mechanics, and consequential pedagogical challenges and instructional errors (Ketelhut et al., 2020; Mouza et al., 2017; Sentance & Csizmadia, 2017).



Participants in the “Absence of Diagnosis” phase could not appropriately deploy core CT processes (e.g., variable-based data handling, control flow, and debugging) because misconceptions and conceptual misunderstandings distorted their conceptual representations, resulting in fragmented and ineffective enactment. Diagnosis-based intervention, by contrast, was marked by participants’ ability to coordinate these processes more coherently, indicating a more integrated understanding of CT concepts and principles. This contrast mirrors the novice–expert divide in CT, where performance depends primarily on the quality of learners’ internal models of how programs and algorithms behave rather than on the execution of isolated procedures. Kohn (2017) documents recurring misconceptions about variable assignment in data processing and updating, including treating assignment as a mathematical equation (e.g., confusion over  $x = x + 1$ ), assuming that a variable can hold multiple values simultaneously rather than only the most recent value, and failing to track how values change across loop iterations. Related work reports difficulties in building accurate mental models of program operation (Ben-Ari, 2001) and in mapping animated/graphical representations onto algorithmic steps (Ben-Bassat Levy et al., 2003), which constrains tracing and obscures how command structure governs control flow. Logical errors, in turn, often stem from misunderstandings of control blocks, conditions, and the expressions embedded in those conditions, producing unexpected output (Alqadi, 2024; Ben-Yaacov & HersHKovitz, 2023; Zhang et al., 2023). Because resolving such errors presupposes advanced problem-solving strategies and a deep grasp of program flow and logic, these breakdowns remain particularly persistent among novices (Alqadi, 2024).

While generalising recurring structures and recognising patterns are foundational for flexible program development, the present study likewise highlights that participants’ difficulties often clustered around pattern recognition, decomposition, and abstraction, namely filtering irrelevant detail to retain essential structure (Boom et al., 2022). Prior research further indicates that when tasks involve integrating multiple algorithmic functions within a single program, novices often experience decomposition breakdowns, including constructing inappropriate block structures, omitting necessary blocks, or compensating by adding superfluous blocks, whereas more proficient approaches typically converge on a more economical block structure that supports efficient program operation (Ben-Yaacov & HersHKovitz, 2023). This approach was also evident in the deep guidance provided by some participants in the present study, who proposed removing unnecessary code blocks when correcting the given PSMP. Debugging adds additional cognitive demands because it requires clarifying the program’s intended behaviour, identifying discrepancies between expected and observed behaviour, generating and evaluating hypotheses about error sources, and implementing and validating fixes (Alqadi, 2024). Debugging has been characterised as one of the most challenging CT practices, and it can remain difficult even for experienced programmers given the complexity of nontrivial programs; nevertheless, prior programming experience may ease debugging (Alqadi, 2024; Shahin et al., 2022).

## Implications for Mathematics Teacher Education and Development

The findings have several implications for the preparation and professional learning of mathematics teachers who integrate CT into mathematics instruction. To begin with, teacher education should develop an integrated understanding of CT processes and mathematical pedagogy. Knowing how to decompose a problem, trace an algorithm, manage variables, or test and debug a program does not necessarily enable teachers to explain these processes or support students who encounter difficulties. This finding is consistent with Maher and Muir (2013), who demonstrated that an instrumental understanding of a mathematical procedure can constrain preservice teachers’ ability to recognise and respond appropriately to students’ errors. Teacher education therefore needs to make explicit the connections between CT principles, mathematical content, and pedagogical decision-making (Mouza et al., 2017; Yadav et al., 2014, 2016).

The findings highlight the importance of developing teachers’ capacity to diagnose student thinking before intervening. The six phases of pedagogical guidance identified in this study suggest that providing a correct answer or a general instruction does not necessarily address the underlying



source of an error. Mathematics teacher education should therefore distinguish between identifying an incorrect response and explaining why the error occurred. This emphasis on interpreting the reasoning underlying students' responses is consistent with research showing that opportunities to reason about students' mathematical strategies can support the development of prospective teachers' professional noticing and knowledge (Tekin-Sitrava et al., 2023). Research on teachers' professional error competence similarly emphasises the importance of identifying the source of an error and selecting appropriate corrective support (Seifried & Wuttke, 2010; Wuttke & Seifried, 2013).

Also, professional learning should provide structured opportunities to practise tracing, testing, and debugging, alongside decomposition, abstraction, and pattern recognition. The findings indicate that difficulties in applying these CT processes were associated with difficulties in diagnosing errors and providing appropriate guidance. Accordingly, professional learning should move beyond introducing CT terminology or familiarising teachers with programming environments. Teachers need repeated opportunities to predict program behaviour, trace changes in variable values, examine control flow, compare intended and actual outputs, and test possible corrections. These activities should be followed by reflection on how the same processes can be used to interpret student thinking and inform instructional responses. Kong et al. (2020) found that sustained participation in programming activities, opportunities to practise teaching, and reflection on pedagogy contributed to teachers' development of CT-related knowledge and practices. The 32-hour professional development experience in the present study also suggests that participation in CT-focused training does not, by itself, ensure that teachers can consistently translate their learning into effective pedagogical guidance. Professional learning should therefore combine CT content knowledge with pedagogical practice, opportunities to apply learning, and reflection on teachers' responses to student errors (Kong et al., 2020; Yadav et al., 2014).

This study illustrated that the distinction between non-diagnostic and diagnosis-based guidance can help teacher educators examine not only whether a response is correct but also whether it accurately identifies the source of an error and provides support that enables students to understand and address it. The six pedagogical guidance phases could inform case-based activities, reflective tasks, and feedback on teachers' responses to simulated student errors. Such approximations of practice provide opportunities for teacher candidates to rehearse responding to errors before encountering the full complexity of classroom teaching (Baldinger & Campbell, 2021). This approach aligns with research that uses incorrect student solutions to examine preservice teachers' capacity to identify misconceptions and explain mathematical reasoning (Korkmaz & Alkan, 2023).

## Conclusion and Limitations

This study is distinguished by its dual focus on tracking the qualitative evolution of pedagogical guidance and quantitatively mapping the appropriate and inappropriate use of CT processes across PSMP-based tasks. By examining the link between teachers' diagnostic capacities and their CT strategy use, the research deepens our theoretical understanding of the competencies required to effectively integrate PSMP-based activities into mathematics instruction. These insights suggest that a planned, structured, and curriculum-aligned integration of CT into education can enhance teachers' ability to diagnose student errors in PSMP tasks while strengthening student-centred approaches that connect key mathematical concepts to real-world applications through CT. The findings also underscore the importance of specialised professional development that strengthens teachers' TPACK for PSMP-based CT instruction, enabling them not only to design and develop PSMP activities but also to provide diagnostic, feedback-rich guidance that supports student self-discovery and the repair of misconceptions in both mathematical concepts and CT principles. In future research and professional learning contexts, the phases should be treated as an analytical framework for examining forms of guidance rather than as a validated, universal sequence through which all teachers necessarily progress. Further research is needed to establish how consistently the phases can be identified across contexts and whether teachers' guidance varies according to the mathematical content, CT process, and task demands.



Restricting the written assessments to 28 participants in the current study—each having previously engaged in a PSMP-based CT education experience—may limit the generalisability of the findings to teachers with diverse CT backgrounds, particularly those without comparable specialised training. Reliance on pre-established theoretical frameworks in DQCA may confine coding to predetermined categories and obscure emergent themes, while exclusive use of written assessments and interview protocols may not capture authentic classroom practice, yielding a somewhat one-dimensional portrayal of teachers' guidance. The absence of a control group, which would enable comparisons across differing levels of CT expertise, further complicates attributing observed outcomes specifically to the PSMP-based CT education. Although the qualitative design captured nuanced intervention behaviours and developmental phases, and the quasi-quantitative components (e.g., frequency counts and coding matrices) offered a snapshot of guidance patterns, a dedicated quantitative study using statistical analyses could sharpen these outcomes and strengthen generalisability.

---

### *Corresponding author*

Feyza Kurban  
Ministry of National Education  
Burdur, Türkiye  
feyza.1526@gmail.com

---

### *Acknowledgements*

The author thanks the academics and instructors who delivered the training and the teachers who participated in the research.

---

### *Funding*

Innovative Educational Practices Support Program (Project No. 123B900).

---

### *Ethical approval*

The study was conducted within the scope of the project, with due attention to ethical considerations. Participation in the research was voluntary, and all participants provided informed consent after being informed of the study's aims and procedures, their right to decline participation or withdraw at any time, and the intended use of the data. Confidentiality was protected through anonymisation of all records and reporting, secure data storage with restricted access, and the use of data solely for research purposes.

---

### *Competing interests*

The author declares no competing interests.

---

## References

- Ahmad, M., & Wilkins, S. (2025). Purposive sampling in qualitative research: A framework for the entire journey. *Quality & Quantity*, 59, 1461–1479. <https://doi.org/10.1007/s11135-024-02022-5>
- Alqadi, B. S. (2024). Enhancing novice programmers' debugging skills through systematic education: A comparative study. *IEEE Access*, 12, 181192–181204. <https://doi.org/10.1109/ACCESS.2024.3509641>
- Assarroudi, A., Heshmati-Nabavi, F., Armat, M., Ebadi, A., & Vaismoradi, M. (2018). Directed qualitative content analysis: The description and elaboration of its underpinning methods and data analysis process. *Journal of Research in Nursing*, 23(1), 42–55.
- Aydeniz, M. (2018). Integrating computational thinking in school curriculum. In M. Khine (Ed.), *Computational thinking in the STEM disciplines* (pp. 253–277). Springer. [https://doi.org/10.1007/978-3-319-93566-9\\_13](https://doi.org/10.1007/978-3-319-93566-9_13)
- Ayres, L. (2008). Semi-structured interview. In *The SAGE encyclopedia of qualitative research methods*. SAGE Publications. <https://doi.org/10.4135/9781412963909.n420>
- Baldinger, E. E., & Campbell, M. P. (2021). Making learning visible: Cases of teacher candidates learning to respond to errors through multiple approximations of practice. *Mathematics Teacher Education and Development*, 23(4), 119–142.
- Barcelos, T. S., Munoz, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics learning through computational thinking activities: A systematic literature review. *Journal of Universal Computer Science*, 24(7), 815–845.
- Barrón-Estrada, M. L., Zatarain-Cabada, R., Romero-Polo, J. A., & Noguez Monroy, J. (2022). Patrony: A mobile application for pattern recognition learning. *Education and Information Technologies*, 27, 1237–1260. <https://doi.org/10.1007/s10639-021-10636-7>
- Beecher, K. (2017). *Computational thinking: A beginner's guide to problem-solving and programming*. BCS, The Chartered Institute for IT.
- Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45–73. <https://www.learntechlib.org/primary/p/8505/>
- Ben-Bassat Levy, R., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, 40(1), 1–15. [https://doi.org/10.1016/S0360-1315\(02\)00076-3](https://doi.org/10.1016/S0360-1315(02)00076-3)
- Ben-Yaacov, A., & Hershkovitz, A. (2023). Types of errors in block programming: Driven by learner, learning environment. *Journal of Educational Computing Research*, 61(1), 178–207. <https://doi.org/10.1177/07356331221102312>
- Boom, K.-D., Bower, M., Siemon, J., & Arguel, A. (2022). Relationships between computational thinking and the quality of computer programs. *Education and Information Technologies*, 27, 8289–8310. <https://doi.org/10.1007/s10639-021-10921-z>
- Bowers, J., Eidin, E., Stephens, L., & Brennan, L. (2023). Examining student testing and debugging within a computational systems modeling context. *Journal of Science Education and Technology*, 32, 607–628. <https://doi.org/10.1007/s10956-023-10049-w>
- Büscher, C. (2025). Differences in students' computational thinking activities when designing an algorithm for drawing plane figures. *International Journal of Science and Mathematics Education*, 23, 365–386. <https://doi.org/10.1007/s10763-024-10465-3>
- Castle, S. D. (2023). Leveraging computational science students' coding strengths for mathematics learning. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education* (Vol. 1, pp. 263–269). Association for Computing Machinery. <https://doi.org/10.1145/3545945.3569861>
- Dahshan, M., & Galanti, T. (2024). Teachers in the loop: Integrating computational thinking and mathematics to build early place value understanding. *Education Sciences*, 14(2), 201. <https://doi.org/10.3390/educsci14020201>
- Dogan, S., Dogan, N. A., & Celik, I. (2021). Teachers' skills to integrate technology in education: Two path models explaining instructional and application software use. *Education and Information Technologies*, 26(3), 1311–1332. <https://doi.org/10.1007/s10639-020-10310-4>
- European Commission, Joint Research Centre. (2022). *Reviewing computational thinking in compulsory education: State of play and practices from computing education*. Publications Office of the European Union. <https://doi.org/10.2760/126955>
- Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12–28. <https://doi.org/10.1002/cae.22255>
- Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5), 487–501. <https://doi.org/10.1080/10494820903520040>



- Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. K. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education*, 17(4), 458–477. <https://www.learntechlib.org/p/174316/>
- Garcia, M. B. (2021). Cooperative learning in computer programming: A quasi-experimental evaluation of jigsaw teaching strategy with novice programmers. *Education and Information Technologies*, 26(4), 4839–4856. <https://doi.org/10.1007/s10639-021-10502-6>
- Garcia, M. B. (2024). Profiling the skill mastery of introductory programming students: A cognitive diagnostic modeling approach. *Education and Information Technologies*. <https://doi.org/10.1007/s10639-024-13039-6>
- Grivokostopoulou, F., Perikos, I., & Hatzilygeroudis, I. (2017). An educational system for learning search algorithms and automatically assessing student performance. *International Journal of Artificial Intelligence in Education*, 27, 207–240. <https://doi.org/10.1007/s40593-016-0116-x>
- Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning in school* (pp. 20–38). Bloomsbury Publishing.
- Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, 4(1), 48–69. <https://doi.org/10.1007/s40751-017-0038-8>
- Hijón-Neira, R., Santacruz-Valencia, L., Pérez-Marin, D., & Gómez-Gómez, M. (2017). An analysis of the current situation of teaching programming in primary education. In *2017 International Symposium on Computers in Education (SIIE)* (pp. 1–6). IEEE. <https://doi.org/10.1109/SIIE.2017.8259650>
- Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of Science Education and Technology*, 29(1), 174–188. <https://doi.org/10.1007/s10956-019-09798-4>
- Kohn, T. (2017, March 8–11). Variable evaluation: An exploration of novice programmers' understanding and common misconceptions. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. Association for Computing Machinery. <https://doi.org/10.1145/3017680.3017724>
- Kong, S. C., Lai, M., & Li, Y. (2023). Scaling up a teacher development programme for sustainable computational thinking education: TPACK surveys, concept tests and primary school visits. *Computers & Education*, 194, Article 104707. <https://doi.org/10.1016/j.compedu.2022.104707>
- Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, 151, Article 103872. <https://doi.org/10.1016/j.compedu.2020.103872>
- Korkmaz, E., & Alkan, S. (2023). The preservice teachers' approaches toward incorrect probability problems. *SAGE Open*, 13(4), 1–15. <https://doi.org/10.1177/21582440231206605>
- Kravik, R., Berg, T. K., & Siddiq, F. (2022). Teachers' understanding of programming and computational thinking in primary education: A critical need for professional development. *Acta Didactica Norden*, 16(4), Article 9, 1–23. <https://doi.org/10.5617/adno.9194>
- Lee, S. W.-Y., Tu, H.-Y., Chen, G.-L., & Lin, H.-M. (2023). Exploring the multifaceted roles of mathematics learning in predicting students' computational thinking competency. *International Journal of STEM Education*, 10, Article 64. <https://doi.org/10.1186/s40594-023-00455-2>
- Léonard, M., Peter, Y., Secq, Y., & Fluckiger, C. (2022). Computational thinking: Focus on pattern identification. In *Educating for a new future: Making sense of technology-enhanced learning adoption: 17th European Conference on Technology Enhanced Learning, EC-TEL 2022, Toulouse, France* (pp. 187–200). Springer. [https://doi.org/10.1007/978-3-031-16290-9\\_14](https://doi.org/10.1007/978-3-031-16290-9_14)
- Maher, N., & Muir, T. (2013). “I know you have to put down a zero, but I'm not sure why”: Exploring the link between pre-service teachers' content and pedagogical content knowledge. *Mathematics Teacher Education and Development*, 15(1), 72–87.
- Maher, C. A., & Sigley, R. (2014). Task-based interviews in mathematics education. In S. Lerman (Ed.), *Encyclopedia of mathematics education*. Springer. [https://doi.org/10.1007/978-94-007-4978-8\\_147](https://doi.org/10.1007/978-94-007-4978-8_147)
- Mayring, P. (2000). Qualitative content analysis. *Forum: Qualitative Social Research*, 1(2), Article 20. <https://doi.org/10.17169/fqs-1.2.1089>
- Ministry of National Education. (2018a). *Matematik dersi öğretim programı (İlkokul ve ortaokul 1, 2, 3, 4, 5, 6, 7 ve 8. sınıflar)* [Mathematics curriculum for primary and lower secondary schools (Grades 1–8)]. <https://mufredat.meb.gov.tr/Dosyalar/201813017165445-MATEMAT%C4%B0K%20%C3%96%C4%9ERET%C4%B0M%20PROGRAMI%202018v.pdf>



- Ministry of National Education. (2018b). *Ortaöğretim matematik dersi (9, 10, 11 ve 12. sınıflar) öğretim programı* [Upper secondary mathematics curriculum (Grades 9–12)]. <https://mufredat.meb.gov.tr/Dosyalar/201821102727101-OGM%20MATEMAT%C4%B0K%20PRG%2020.01.2018.pdf>
- Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, 108(6), 1017–1054. <https://doi.org/10.1111/j.1467-9620.2006.00684.x>
- Mohottige, N. U. S. R., Bjerke, A. H., & Andersen, R. (2024). Teacher education as stakeholder: Teacher educator perspectives on the integration of computational thinking into mathematics and science courses. *Teachers and Teaching*, 30(4), 437–451. <https://doi.org/10.1080/13540602.2024.2313635>
- Mouza, C., Yang, H., Pan, Y.-C., & Ozden, S. Y. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, 33(3), 61–76. <https://doi.org/10.14742/ajet.3521>
- Palinkas, L. A., Horwitz, S. M., Green, C. A., Wisdom, J. P., Duan, N., & Hoagwood, K. (2015). Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *Administration and Policy in Mental Health and Mental Health Services Research*, 42(5), 533–544. <https://doi.org/10.1007/s10488-013-0528-y>
- Panselinas, G., Mavrikis, M., Geraniou, E., Anastasiou, P., Tambouratzis, E., & Kartsonakis, E. (2020). Towards a teaching model for managing students' cognitive load in activities that integrate computational and mathematical thinking. *European Journal of Engineering and Technology Research: Special Issue: Conference on Informatics in Education 2019* (pp. 6–11). <https://doi.org/10.24018/ejers.2020.0.cie.1792>
- Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1, 95–123. <https://doi.org/10.1007/BF00191473>
- Rich, P. J., Egan, G., & Ellsworth, J. (2019, July 15–17). A framework for decomposition in computational thinking. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)* (pp. 1–6). Association for Computing Machinery. <https://doi.org/10.1145/3304221.3319793>
- Rycroft-Smith, L., & Connolly, C. (2019). Comparing conceptions of mathematical and computational thinking cycles. *Cambridge Mathematics*, 29, 1–2.
- Saritepeci, M. (2022). Modelling the effect of TPACK and computational thinking on classroom management in technology enriched courses. *Technology, Knowledge and Learning*, 27(4), 1155–1169. <https://doi.org/10.1007/s10758-021-09529-y>
- Seifried, J., & Wuttke, E. (2010). Student errors: How teachers diagnose and respond to them. *Empirical Research in Vocational Education and Training*, 2(2), 147–162. <https://doi.org/10.25656/01:8250>
- Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2), 469–495. <https://doi.org/10.1007/s10639-016-9482-0>
- Sentance, S., Dorling, M., & McNicol, A. (2013). Computer science in secondary schools in the UK: Ways to empower teachers. In I. Diethelm & R. Mittermeir (Eds.), *Informatics in schools: Sustainable informatics education for pupils of all ages* (pp. 15–30). Springer-Verlag.
- Shaffer, C. A., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce, S., & Edwards, S. H. (2010). Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 10(3), Article 9. <https://doi.org/10.1145/1821996.1821997>
- Shahin, M., Gonsalvez, C., Whittle, J., Chen, C., Li, L., & Xia, X. (2022). How secondary school girls perceive computational thinking practices through collaborative programming with the micro:bit. *Journal of Systems and Software*, 183, Article 111107. <https://doi.org/10.1016/j.jss.2021.111107>
- Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms. *The Science Teacher*, 81(5), 53.
- Sun, C., Yang, S., & Becker, B. (2024). Debugging in computational thinking: A meta-analysis on the effects of interventions on debugging skills. *Journal of Educational Computing Research*, 62(4), 1087–1121. <https://doi.org/10.1177/07356331241227793>
- Sung, W., & Black, J. B. (2021). Factors to consider when designing effective learning: Infusing computational thinking in mathematics to support thinking–doing. *Journal of Research on Technology in Education*, 53(4), 404–426. <https://doi.org/10.1080/15391523.2020.1784066>
- Tekin-Sitrava, R., Işıksal-Bostan, M., & Çatman-Aksoy, E. (2023). The interplay between professional noticing and knowledge: The case of whole number multiplication. *Mathematics Teacher Education and Development*, 25(2), Article 5.
- Thompson, D., & Bell, T. (2013). Adoption of new computer science high school standards by New Zealand teachers. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSE '13)* (pp. 87–90). Association for Computing Machinery. <https://doi.org/10.1145/2532748.2532759>



- Thompson, D., Bell, T., Andreae, P., & Robins, A. (2013). The role of teachers in implementing curriculum changes. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. Association for Computing Machinery.
- Vourletsis, I., Politis, P., & Karasavvidis, I. (2021). The effect of a computational thinking instructional intervention on students' debugging proficiency level and strategy use. In T. Tsiatsos, S. Demetriadis, A. Mikropoulos, & V. Dagdilelis (Eds.), *Research on E-Learning and ICT in Education* (pp. 15–34). Springer International Publishing. [https://doi.org/10.1007/978-3-030-64363-8\\_2](https://doi.org/10.1007/978-3-030-64363-8_2)
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25, 127–147. <https://doi.org/10.1007/s10956-015-9581-5>
- Weintrop, D., Rutstein, D., Bienkowski, M., & McGee, S. (2021). Assessment of computational thinking. In A. Yadav & U. Berthelsen (Eds.), *Computational thinking in education: A pedagogical perspective* (pp. 90–111). Routledge. <https://doi.org/10.4324/9781003102991-6>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>
- Wuttke, E., & Seifried, J. (2013). Diagnostic competence of (prospective) teachers in vocational education: An analysis of error identification in accounting lessons. In K. Beck & O. Zlatkin-Troitschanskaia (Eds.), *From diagnostics to learning success: Proceedings in vocational education and training* (pp. 225–240). Sense Publishers.
- Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K–12 classrooms. *TechTrends*, 60(6), 565–568. <https://doi.org/10.1007/s11528-016-0087-7>
- Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, 60(4), 55–62. <https://doi.org/10.1145/2994591>
- Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2014). Introducing computational thinking in education courses. *ACM Transactions on Computing Education*, 14(1), 1–16. <https://doi.org/10.1145/2576872>
- Yamashita, K., Fujioka, R., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2016). Practices of algorithm education based on discovery learning using a program visualization system. *Research and Practice in Technology Enhanced Learning*, 11, Article 15. <https://doi.org/10.1186/s41039-016-0041-5>
- Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K–12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, 10, Article 3. <https://doi.org/10.1186/s40594-023-00396-w>
- Zhang, Y., Paquette, L., Pinto, J. D., Liu, Q., & Fan, A. X. (2023). Combining latent profile analysis and programming traces to understand novices' differences in debugging. *Education and Information Technologies*, 28(4), 4673–4701. <https://doi.org/10.1007/s10639-022-11343-7>
- Zhang, Y., & Wildemuth, B. M. (2009). Qualitative analysis of content. In B. M. Wildemuth (Ed.), *Applications of social research methods to questions in information and library science* (pp. 308–319). Libraries Unlimited.

