Designing Feedback for Exploratory Mathematical Activities: A Computational Thinking Perspective

Computational Thinking Scaffolding for Exploratory Mathematical Activities

Filiz Mumcu[1], Manolis Mavrikis[2], Sokratis Karkalas[3], Christina Gkreka[4], Charikleia Korompli[4], Dimitris Diamantidis[4], and Zsolt Lavicza[1]

[1] Johannes Kepler University
[2] University College London, UCL Knowledge Lab
[3] University of Derby
[4] National and Kapodistrian University of Athens

Abstract

Computational thinking (CT), recognised as vital in mathematics education, closely aligns with problem-solving and is increasingly integrated into global curricula. In this context, designing and engaging learners in exploratory activities through programming and debugging is valuable. However, challenges remain in effectively incorporating CT into teaching, especially in designing and assessing such exploratory activities. This study explores these issues by developing and evaluating a prototype incorporating CT as a framework for feedback design in exploratory mathematical activities involving programming. The prototype employs the MaLT2 programming environment, inspired by Papert's turtle geometry, within AuthELO, an authorable feedback design system. MaLT2 aids mathematical understanding through programming, while AuthELO offers real-time, context-aware feedback based on students' interactions. We design a feedback mechanism that applies CT components to guide learners in open-ended tasks. Qualitative data from a semi-structured focus group with educators reveals that they appreciate CT-based feedback for its relevance to math and programming, though they struggle with abstraction. Participants indicated a need for specific and motivating feedback, emphasising the balance between guidance and exploratory learning. The study highlights the importance of authorable feedback systems facilitating iterative refinement of exploratory mathematics programming activities with CT as a scaffolding framework.

Keywords and Phrases: Exploratory mathematical activities, Feedback design, Computational thinking

1. Introduction

Exploratory learning environments (ELEs) differ from traditional ones in that they are open-ended and promote constructivist learning. They support exploration, allowing students to vary parameters and observe model effects. Research shows ELEs

Mumcu, F., Mavrikis, M., Karkalas, S., Gkreka, C., Korompli, C., Diamantidis, D., & Lavicza, Z. (2025). Designing Feedback for Exploratory Mathematical Activities: A Computational Thinking Perspective. Computational Thinking Scaffolding for Exploratory Mathematical Activities. *Constructionism Conference Proceedings, 8/2025*, 285–300. https://doi.org/10.21240/constr/2025/78.X

enhance learning outcomes (Van Joolingen & Zacharia, 2009; Noss & Hoyles, 1996) through a rich, student-centered experience. Modern educational software reflects this flexibility, promoting exploration and trial and error (Amir & Gal, 2013). Their self-directed nature requires less teacher involvement, making them suitable for large classes or resource-limited contexts (Gal et al., 2008; Pawar et al., 2006), especially if orchestration challenges are addressed (Mavrikis et al, 2019).

Since Papert's *Mindstorms (1980)*, extensive research has been conducted on how programming can be a powerful tool for students to explore mathematical concepts. The programming activity is inherently mathematical (Feurzeig et al., 2011), involving problem-solving, logical thinking, decomposition, and debugging. LOGO, for example, was designed to implement these ideas, providing a natural framework and vocabulary for discussing mathematics (Feurzeig et al., 2011) in exploratory "microworlds." Papert (1980) proposed that students' mathematical reasoning could be enhanced by testing and debugging their solutions through programming.

Over the past two decades, computational thinking (CT) has increasingly been recognised as a valuable skill across various subjects, particularly in mathematics education (Mumcu et al., 2023a; Refvik & Bjerke, 2022; Turgut et al., 2024). CT involves deconstructing a problem into smaller components, filtering out irrelevant information, identifying recurring patterns, and formulating a solution that any machine can interpret (Weintrop et al., 2021; Wing, 2006). These skills are closely linked to problem-solving (Voogt et al., 2015; Yadav et al., 2011). Across the globe, digital education curricula are undergoing significant changes to emphasise the development of CT and problem-solving skills (Kastner-Hauler et al., 2025).

While it is possible to develop CT skills within mathematics education, it is also feasible to enhance mathematical thinking through CT skills (Bråting & Kilhamn, 2021; Refvik & Bjerke, 2022; Rich et al., 2020). CT and mathematics education share fundamental problem-solving foundations, though they emphasise different aspects. While CT emphasises computational approaches like decomposition and algorithmic thinking (Wing, 2006), mathematical problem-solving has long been structured around understanding, planning, executing and reflecting (Polya, 1945). CT possesses scaffolding power for problem-solving (Kale et al., 2018; Lee et al., 2024). Engaging in hands-on programming activities particularly deepens students' conceptualisation of mathematical concepts and problem-solving (Ng & Cui, 2021). However, incorporating CT into the teaching of a subject is an area where teachers generally feel unprepared, particularly regarding instructional design and the assessment of student thinking (Boulden et al., 2021; Mumcu et al., 2023b; Turgut et al., 2024). Furthermore, students face various challenges in programming environments, where they must simultaneously tackle mathematical problems while engaging in programming (Cui & Ng, 2021).

Supporting learners in ELEs requires extensive preparation, skill, and continuous monitoring of the learning process. Teachers must track and understand students' behaviours and how they construct knowledge (Morales Gamboa & Gamboa, 2000). Since problems in ELEs often lack clear boundaries between correct and incorrect approaches, providing effective guidance is challenging, time-consuming, and resource-intensive (Gutierrez-Santos et al., 2012). It can be challenging for teachers to monitor progress and assess performance in this open-ended context (Amir & Gal, 2013). Research shows that too much freedom without adequate support can lead to

distraction, loss of focus, and hinder learning outcomes (Mayer, 2004; Kirschner et al., 2006; Klahr & Nigam, 2004). Proper guidance is essential to ensure effective learning in ELEs (Kirschner et al., 2006).

In this study, we design, develop, and evaluate a prototype that utilises CT as a scaffolding framework for constructing authorable feedback for exploratory mathematical activities involving programming. The study examines how CT-based feedback design could be tailored for exploratory mathematical activities. Based on this research problem, the study seeks answers to the research questions outlined below:

1. How do educators perceive and utilise feedback based on computational thinking framework when engaging in exploratory mathematical activities through programming?
2. How do they implement this feedback during the exploratory mathematical activity?
3. What improvements do educators suggest to improve the feedback mechanisms to support their learning?

2. Related Work

2.1 Exploratory microworlds for mathematics education

"Microworlds" are computational environments incorporating scientific concepts and their relationships, designed to engage students in exploration and constructive activities (Healy & Kynigos, 2010), enabling them to transfer inquiry habits from their lives into constructing scientific meaning (Papert, 1980). "Half-baked" microworlds are fallible artefacts intentionally designed with bugs (Kynigos, 2007), prompting users to explore, modify, and question underlying rules and concepts. These microworlds serve as starting points for critical thinking and idea generation, promoting malleability and deep structural access for building or decomposing software. The goal is to foster engagement and problem-solving, allowing students to practice computational skills like debugging through direct interaction with the environment.

Several studies since then employed LOGO-like microworlds to examine meaning generation about several mathematical concepts like angles (Mitchelmore & White, 2000), trigonometric functions (Kynigos & Karavakou, 2023) and geometrical shapes properties (Clements et. al., 2001), concluding that these environments foster students' understanding of mathematics, allowing them at the same time to engage naturally in CT practices like problem decomposition (Wing, 2006), algorithmic thinking (Aho, 2012), and iterative refinement through testing and debugging (Grover & Pea, 2013). Rich opportunities do not guarantee learning (Balacheff & Kaput, 1996). Early studies show students often rely on visual shapes instead of thinking mathematically when programming with Logo (Clements & Meredith, 1993). Their actions may disconnect from the mathematical concepts (Geraniou et al., 2010). They also struggle with CT concepts like abstraction and debugging, hindering mathematical problem-solving (Brennan & Resnick, 2012).

In this context, teachers are charged with providing extra pedagogical support, ensuring that their students are explicitly aware of the strategies and processes they follow to learn and provide individualised feedback regarding their efforts (Clements & Meredith, 1993). Geraniou et al. (2010) showed that during exploratory programming activities, students needed significant support to stay focused towards explicit goals. To achieve that, teachers must employ several pedagogical support mechanisms,

including provoking cognitive conflict, encouraging alternative solutions, supporting reflection, and promoting motivation (Mavrikis et al., 2008). At the same time, several studies highlight that even teachers lack a clear understanding of how programming and CT could be effectively integrated into educational practices (e.g., Denning, 2017; Grover & Pea, 2013; Mumcu et al., 2025; Yadav et al., 2017).

This study first explores how CT can facilitate educators' engagement in exploratory mathematical activities through programming. Second, it investigates how CT can support them in authoring meaningful feedback to their students, ensuring adequate support is provided while working with programming-based exploratory learning activities.

2.2 Authorable feedback in exploratory learning activities

Developing automated support is a resource-intensive process, often requiring significant technical expertise from domain experts, knowledge engineers, and specialised developers. Additionally, much of this development typically needs to be completed prior to the system's release to end users. Efforts have been made to reduce technical barriers through systems that combine existing materials with teaching expertise (Blessing et al., 2007; Ainsworth et al., 2003; Aleven et al., 2009). However, these systems are often limited, domain-specific, and primarily support guided learning rather than fostering ELEs.

Research on authoring tools for ELEs is limited, but some efforts have focused on supporting teachers and students. Amir & Gal (2013) used plan recognition to visualise student activities, while Pearce-Lazard et al. (2010) and Gutierrez-Santos et al. (2012) enhanced teacher support in MiGen through tracking and feedback. Cocea & Magoulas (2010) explored group formation based on learner strategies, and Cocea et al. (2008, 2009a, 2009b) proposed modeling and feedback mechanisms. Bunt et al. (2004) emphasised meta-cognitive skills, while machine learning approaches (Amershi & Conati, 2006; Bernardini & Conati, 2010) have been used for adaptive support. Despite these contributions, challenges remain. The authoring process requires significant domain knowledge and technical expertise, leading to domain-specific solutions with limited applicability across diverse learning tools.

AuthELO (Karkalas et al., 2016; 2017) addresses the domain specificity problem by drawing inspiration from example-tracing tutors (Aleven et al., 2009), where authors generate feedback by performing tasks as students. This creates a tree-like representation of student states; which authors can annotate to define tutor behavior. However, due to their open-ended nature, traditional example-tracing is domain-specific and unsuitable for ELEs. To overcome this, AuthELO avoids visualising all possible states and instead relies on student interaction data to define task-dependent feedback rules, making it more adaptable. Conceptually and architecturally, it resembles SEPIA (Ginon et al., 2014), an external support system that integrates with learning tools without modifying them. By not relying on domain-specific models, AuthELO offers a flexible, domain-independent solution for automated feedback.

3. Methodology

3.1 Research design and participants

In this study, we developed a new prototype by integrating the AuthELO (authorable feedback design app) into the MaLT2 (MachineLab Turtleworlds 2) programming environment following Type 1 developmental research phases (planning, conducting, and reporting), as Richey and Klein (2005) outlined. Seven educational researchers participated in the study. One has a PhD in computer science education (coded as C), five PhD students in STEM education (coded as S1, S2, ...) and one graduate student in math and computer science education (coded as M). All of them have teaching experience. This composition was intentional to capture varied perspectives on CT-integrated feedback. When engaging with the prototype, the computer science education expert likely focused on programming logic and debugging efficiency. STEM education researchers may have prioritised pedagogical implications, such as how feedback scaffolds mathematical reasoning. The math and computer science education researcher potentially bridged these perspectives, assessing both mathematical validity and computational implementation. We asked participants to consciously adopt a 'learner' mindset when evaluating the prototype.

3.2 Data collection and analysis

The qualitative data was collected through a semi-structured focus group interview. The interview questions are designed to delve deeper into educators' experiences and perceptions regarding feedback design and its impact on their engagement through exploratory math activities (see Appendix). First, an hour-long interview recording was transcribed verbatim. Data was analysed using content analysis. We utilised thematic analysis using an inductive approach to examine the focus group data. Following Miles and Huberman's (1994) framework, we performed open coding to identify emerging patterns, organised codes into meaningful themes through iterative analysis, and verified interpretations through researcher triangulation.

4. Workshop Design and Implementation

4.1 The design of the workshop

Participants attended a 120-minute capacity-building session led by our research team, offering an in-depth overview of key concepts in CT. We discussed what CT means, core CT components such as data processing, analysis, representation, abstraction, decomposition, pattern recognition, algorithm design, debugging/testing, and the connections between CT and math, focusing on real-world problems. The session featured unplugged computer science activities and used Bebras (https://www.bebras.org/) challenge questions, which participants tackled and discussed to enrich their grasp of CT.

Building on this foundation, the workshop was structured to progressively develop participants' understanding and application of CT concepts. After the initial capacity-building session, participants worked individually on the exploratory activity within the MaLT2 environment, applying their newfound knowledge. Participants were also asked to put themselves in the students' shoes during this application and consider what challenges students might face in completing the assigned tasks. By engaging as learners, educators experienced firsthand the cognitive and technical

challenges students might face when interacting with the CT-integrated feedback system. The workshop concluded with a focus group interview, providing a platform for participants to discuss their experiences regarding the feedback.

4.2 Materials: MaLT2 and AuthELO

The MaLT2 environment is a LOGO-like system inspired by Papert's "turtle geometry" approach. It aims to support students' mathematical meaning generation, by making constructions through programming, by integrating textual programming with the affordances of dynamic manipulation of parametric procedures, 3D graphics and camera navigation. MaLT2 includes the following interrelated representations: *i. Symbolic (code), ii. Figural* and iii. *Dynamic* manipulation of generalised models controlled by procedures with variables (Kynigos & Grizioti, 2018). It allows students to visualise their coding processes via a turtle graphics interface, effectively bridging programming with mathematics. To optimise the use of the MaLT2 exploratory learning environment (microworld), a feedback mechanism was integrated via the AuthELO platform. AuthELO is an authorable feedback design system that provides real-time, context-aware assistance to learners interacting with microworlds (Karkalas, Mavrikis, 2016).

The prototype uniquely integrates AuthELO into MaLT2, offering a feedback mechanism that aids student progress in challenging open-ended activities. As students use MaLT2, AuthELO logs their actions, tracking commands, slider usage, and camera controls to provide personalised feedback based on their interactions. This data delivers targeted guidance throughout their learning. The architecture supports enhancing MaLT2 or any digital learning environment with adaptable feedback design via AuthELO, generating components that enhance functionality noninvasively. This integration empowers educators to create effective adaptive learning experiences, monitoring individual student progress and learning journeys.

CT-integrated feedback was collaboratively designed by the researchers. The prototype provides tailored feedback based on user movements in MaLT2, using CT components as a scaffolding framework (see Table 1). This method aligns feedback with key CT components: decomposition, pattern recognition, abstraction, algorithm design, and debugging. It supports users in understanding mathematical concepts and promotes computational thinking. This approach streamlines teaching and enhances learning experiences, empowering students with guidance and reinforcing CT skills.

Feedback Message	Trigger Condition	CT component
Have you tried moving the sliders? Clicking on the trace and moving the sliders might help you understand how the parameters change the result.	The sliders have not been moved	debugging
Make sure to keep the height and width of the rectangle as parameters to keep the procedure re-usable.	There are fewer than 2 variables in the procedure definition	abstraction
Have you tried breaking down the problem into small manageable steps?	bottom-out (randomly, if no condition applies)	decomposition
Are there any recurring similarities or trends for solving the problem?	bottom-out (randomly, if no condition applies)	pattern recognition

Feedback Message	Trigger Condition	CT component
What does the procedure do when you change the values of a and b and what does this tell you about the procedure?	bottom-out (randomly, if no condition applies)	abstraction, pattern recognition
Are there any changes you can try out to get a better solution to the problem?	bottom-out (randomly, if no condition applies)	debugging
The rectangle procedure accepts 2 parameters. Can you pretend to be the sparrow and draw the path it takes on a sheet of paper? In what order must the variables appear to form a proper rectangle without the lines sticking out in one corner?	The 2 variables in the rectangle procedure do not alternate	decomposition, algorithm design

Table 1: Examples from CT-integrated feedback design with AuthELO.

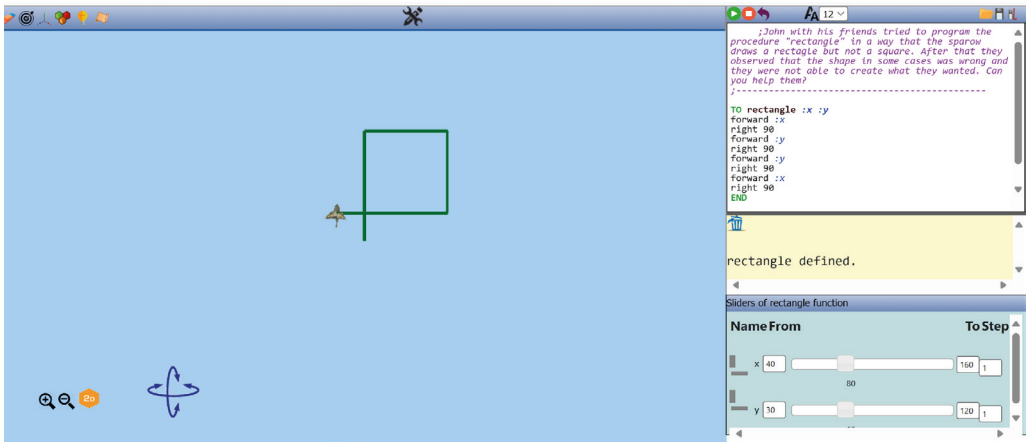

Figure 1: The exploratory math activity designed in the MaLT2 programming environment (http://etl.ppp.uoa.gr/malt2/).

4.3 *Implementation of the exploratory math activity*

In this study, exploratory math activity is described as a learner-centred task that combines mathematical reasoning with programming in an open-ended environment, allowing participants to investigate, test hypotheses, and refine solutions through experimentation. For this purpose, participants were given a task which included a faulty procedure designed to draw a rectangle, applying algebraic symbols, relationships, and the properties of squares and rectangles to fix it (see Figure 1). The story behind the activity was that: "John, with his friends, tried to program the procedure "rectangle" in a way that allowed the sparrow to draw a rectangle but not a square. After that, they observed that the shape, in some cases, was wrong and that they could not create what they wanted. Can you help them?"

5. Results

5.1 *How do educators perceive and utilise CT-integrated feedback when engaging in exploratory mathematical activities through programming?*

Educators expressed that when engaging in exploratory mathematical activities through programming, they felt that CT-integrated feedback included the components of CT, such as algorithm design, debugging, and testing, but not abstraction.

> "I felt, for example, debugging and testing phases. I felt the feedbacks. ... Especially like this, algorithm design. Algorithm design, debugging and testing. Abstraction and I need to get some feedback about abstraction, for example, but I couldn't." (S2)

This limitation was corroborated by S3, suggesting the feedback system could be strengthened by more comprehensive coverage of all CT components.

> "I agree. I didn't find feedback for each step. I found feedback only in these steps." (S3)

Furthermore, the feedback informed educators about the strategies they developed for solving the problem. One of the participants underlined that the feedback on the strategies could increase students' confidence. This highlights how targeted feedback on problem-solving approaches, beyond just correctness, can support the learning process.

> "I prefer to get feedback on my strategies, whether it is correct or not, whether I have to move on to the next activities or not, because it will build my confidence as a student. So it's important to comment, or inform me, or relate it to my strategies." (S1)

In addition to recognising the strong relationship between mathematics, programming, and CT, educators also highlighted the challenges of integrating programming into mathematics education.

> "They are related. We cannot program without math. Also, we can develop our math skills and CT skills by programming. We can use also programming for math skills. We see in this exercise, they are related." (C)

However, S5 offered a contrasting classroom perspective; these differing views underscore the importance of accessibility when integrating programming into math education.

> "I have the opposite experience. People who aren't interested in math won't even think about doing programming because they hate it so much. Because they don't get the patterns, because they are so strict. It's a very strict system. So I like to approach at the beginning with showing them they already know how to do all those things. Because it's the human way of thinking, they aren't just in their conscience to think about how they do cooking or more routines and so on." (S5)

Educators indicated they needed more preparation and resources to integrate programming into mathematics education. CT-integrated feedback can play a crucial role in helping teachers effectively implement exploratory tasks in the classroom. By providing insights into problem-solving strategies and offering guidance on key components like algorithm design, debugging, and testing, educators can feel more confident and better prepared to integrate programming into their mathematics teaching.

> "And then, you brought to provide a lot of examples on the other hand. So it would be really beneficial if we deal with new innovation in educational setting. Teachers need example and resources. Not only example of the task, but also how to implement it. Let's say the sample of lesson plans. It's also useful based on teachers' responses previous. And then, I've learned also about

> guidelines for teachers to develop that kind of tasks. Maybe there are main principles or key principles that is easier to be followed by teachers because they don't have time to read the theories of CT." (S4)

Educators underlined the difficulties of implementing CT teaching activities and stated that CT-integrated feedback would facilitate them. The challenge of implementation was summarised by S2, revealing anxiety about adopting these new approaches.

> "So for me as a teacher, I think somehow it's challenging, or I can say scary. Why? Because I consider about the task." (S2)

5.2 How do they utilise and implement this feedback during the exploratory math activity?

The feedback helped educators evolve from using simple commands to more complex ones, showing its effectiveness in scaffolding learning. S3 illustrates how feedback can guide learners toward more sophisticated approaches.

> "Because at the beginning, I only used the simple command. I mean, I used a square of 3,200, because I think about the 45, one, one, and one square of two. And I changed the length of the side. Afterward, I realised there is a command of multiplication, and then I create another variable. So I create a new variable Z, and then I create a new procedure. So it's suddenly significantly changing after, in detail, learning about the command. So I start from simple, and then I create another command." (S3)

Exposure to new coding platforms, supported by feedback, was appreciated by educators, as it enhanced their understanding regarding how to program:

> "Yes, I tried before some coding in another platform. And for me, it's always hard to remember this sentence that you need to write at the beginning and at the end of the coding. So the feedback was useful for me in this way, because sometimes I check what we are expecting or something like that. And I like trying different platforms to code. So all of them are working in a different way. So I think it was helpful in this way, just to know a new platform for me." (M)

Feedback influenced how educators decomposed problems, leading to a more structured approach to problem-solving, as S1 and S2 reflected:

> "I didn't know how I decomposed the problem. Previously, I thought it was complicated or tried to work with something simple, and then I tried to. I've done with this; let's do with another, maybe complicated or challenging, if it is possible." (S1)

> "I've got a feedback to redefine my function. The programming itself is correct, but when you are troubleshooting, like the rectangle, remember, it's not." (S2)

There was a need for more instant feedback when procedural errors occurred, as educators found it frustrating to run blocks first before receiving feedback. One critique noted by S1:

> "For me, my problem was that, I don't know if the programmers can change this, but the problem for me is why do you have to run the code first, like the block first, before calling it? Because earlier, when I changed the code, I know

> that I'm correct. But when I click on the play, on the run button, it's wrong. And I think later when I realise that, oh, I have to run this again. So I think there should also be a feedback. Or maybe, I don't know, because the computer just ran the previously run code. So I think there's no feedback on that." (S1)

5.3 What improvements do educators suggest to improve the feedback mechanisms to support their learning?

Educators emphasised the importance of feedback being both specific to the task and easily readable. As tasks became more complex, feedback often lost its specificity and became harder to read, particularly when multiple procedures were involved. This lack of clarity made it difficult for educators to understand what they needed to improve. S2 illustrated this need, and S3 expanded on this by describing a need for contextual guidance:

> "It has to be specific... a more specific feedback would be what is the property of the length of the triangle? So what is the relationship among the lengths of the triangle? So I think it's more specific than saying are there any changes that you can try out? So it has to be like specific." (S2)

> "I have a difficulty, for example, I am drawing the triangle, and my friend helped me. I couldn't find the useful feedback before I am drawing the triangle, for example. Maybe it can be given feedback, for example, please go to the point which you start the triangle, and maybe it can give me some feedback about the angle, for example, angles. I need like this feedback." (S3)

Beyond cognitive support, educators stressed the importance of feedback's affective dimension, the need for motivational feedback. S1 explained how confirmation of correct approaches builds confidence, and S5 further emphasised feedback's role in validation:

> "One thing more, for example, when you are testing, you don't receive feedback if you are doing it in a correct way. You never say, get well done, or this is correct, or something, and I think as a student it's important just to know that I finished the task. I don't need to call the teacher all the time, and it's really motivating for me." (S1)

> "Positive feedback, for example." (S4)

> "For me, feedback is about recognising someone or somebody. I think that's the most important part for students. Because they feel recognised. That's why they need feedback. And that's why it would be important to make it more sufficient or more on the point of what they were doing than providing feedback for it." (S5)

Educators also suggested system-level improvements to support the feedback experience and noted the feasibility of improving feedback mechanisms. M recommended how alternative presentation formats could make complex feedback more accessible.

> "Feasible. I think, for example, on some platforms, I've seen animation or silent video feedback to correct the answer. Here, it's a bit hard for me to follow the feedback. I found the yellow box at the bottom, but it didn't really help me. So it is possible to make some animation or silent video to give feedback. You need to click this one and that one. I think the feasibility aspect is important." (M)

Multiple educators identified visibility and design issues with the current feedback display. S3 noted contrast problems, while S4 described more systemic timing and placement issues. The visual presentation of feedback needed refinement to match the effectiveness of the command-based guidance.

> "The yellow feedback was often not readable... Maybe you can make it red to be more noticeable." (S2)

> "I found that I realised the feedback related to the CT activities, it was late for me, because my focus is on the avatar, and then only typing the procedure, and then look at it moving, and then afterwards, I was like, oh, so I cannot turn this thing back. So it would be better to make it bigger, or move it to the yellow square below the procedure. So it was late for me, because it looked like redefined. But suddenly I realised, oh, there is another coming." (S4)

> "The same happened with me. It happened to me when I was exploring. The same, it's not the entity." (S1)

> "Maybe it's because the colour is yellow. It's not really eye-catching. Maybe you can make it red." (S3)

Feedback should be concise, avoiding lengthy explanations that can overwhelm students. C articulated it clearly:

> "I need feedback that is not really long. Sometimes, the description of the feedback in the platform was really long, so you needed to read a lot just to get the proper idea. So maybe something that is clearer, like check that, for example." (C)

Beyond feedback mechanisms, participants suggested improvements to the broader system interface. S4 identified a workflow challenge, pointing to the need for more flexible navigation controls that better support iterative testing and debugging processes:

> "One idea maybe, I had done the rectangle and was moving forward to the roof, but I didn't have an opportunity to always start again with the roof. That's why I asked about the stop button because for me it was stopping the procedure, but it always went back to the beginning, and that's why I didn't get why we have clear the grid as well, because I had it double, so maybe you could think about redefining the stop button." (S4)

6. Discussion

The study examines how CT-based feedback design could be tailored for exploratory mathematical activities through programming, using CT as a scaffolding framework for constructing authorable feedback. The participants acknowledged the feedback concerning CT components, such as algorithm design, debugging, and testing; however, they paid little attention to abstraction, which may reflect their level of understanding of all CT components. Their observation is both noteworthy and, to a certain extent, anticipated (Brennan & Resnick, 2012; Geraniou et al., 2010). Abstraction is a complex concept that can be difficult to understand in its various forms and interpretations. Participants may not grasp its diverse dimensions, especially in CT-integrated feedback. The Logo language isn't object-oriented and lacks features

like abstract classes and polymorphism, but it allows function definitions, demonstrating abstraction. In AuthELO, functions exemplify abstraction, aligning with MaLT2's capabilities and supported by data.

Participants saw the connection between math, programming, and CT, noting challenges in integrating programming into math education. This points to a common issue in programming education: finding the right granularity for problem-solving in code (Chao, 2016; Cui & Ng, 2021; Kaufmann & Stenseth, 2021). No one-size-fits-all solution exists; the programming language and abstraction level must align with learning goals. Feedback incorporating CT suggests that a higher abstraction level is usually preferable. An ideal high-level language abstracts irrelevant implementation details. This raises a question: *Do participants' discomforts stem from the programming language or from past experiences that fostered misconceptions about coding?* Ultimately, the right abstraction level simplifies tasks rather than complicating them.

Educators need more resources to integrate programming into math education (Boulden et al., 2021; Turgut et al., 2024). In constructionist settings like MaLT2, inquiry-based learning enables independent exploration under tutor supervision. This demands tutors to plan, facilitate, and offer personalised guidance for misconceptions, relying on their understanding of student knowledge and past activities. Tailored support is complex and requires expertise, as tutors can't tackle these challenges alone. CT-integrated feedback is crucial for teachers to implement exploratory tasks effectively. Educators emphasise specific feedback, balancing support with learner independence. Support should assist progress without being overly intrusive. While theory promotes this balance, practical definition is challenging, highlighting the need for adaptable systems that modify help based on student progress, understanding, or task complexity. Educators also highlighted the need for motivational feedback, but automated responses can clash with nuanced, user-driven methods. Therefore, educators should manage engagement strategies to match their class context.

CT-integrated feedback was valuable. However, our analysis showed that design elements in MaLT2 also addressed challenges in visualisations, interactivity, and interface design. These elements work alongside feedback mechanisms, suggesting customisable workspace layouts for diverse problem-solving styles.

7. Conclusion

The findings of this study highlight the potential role of *instant, task-specific feedback* during exploratory mathematical activities involving programming. Educators emphasised that feedback should be *precise, concise, and directly relevant to the task* to enhance comprehension and guide learning effectively. Additionally, there is a clear demand for *motivational feedback* to foster positive emotional engagement and sustain learners' interest. *Practical feedback mechanisms*, such as animations or silent video feedback, can make feedback more accessible and less intrusive. Equally important are the *visibility and design of feedback*; it must be visually distinct and compelling to ensure learners notice and engage with it during activities. By addressing these elements, feedback mechanisms can enhance the effectiveness of exploratory learning environments, supporting the learning process's cognitive and emotional aspects. This study underscores the importance of designing adaptive, context-aware feedback systems that balance guidance with learner autonomy, ultimately enriching the educational experience in exploratory learning environments involving programming.

Acknowledgements

This work was supported by the TransEET Project, funded by the European Union HORIZON-WIDERA -2021- ACCESS -03-01 with Grant no: 101078875.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835. DOI: 10.1093/comjnl/bxs074

Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., & Wood, D. (2003). Redeem: Simple intelligent tutoring systems from usable tools. In *Authoring tools for advanced technology learning environments* (pp. 205–232). Springer.

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education, 19*(2), 105–154.

Amershi, S., & Conati, C. (2006). Automatic recognition of learner groups in exploratory learning environments. *In International Conference on Intelligent Tutoring Systems* (pp. 463–472). Springer.

Amir, O., & Gal, Y. (2013). Plan recognition and visualisation in exploratory learning environments. *ACM Transactions on Interactive Intelligent Systems (TiiS, 3)(3)*, 1–23.

Balacheff, N., & Kaput, J. J. (1996). Computer-based learning environments in mathematics. In *International Handbook of Mathematics Education: Part 1* (pp. 469-501). Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-94-009-1465-0_14

Bernardini, A., & Conati, C. (2010). Discovering and recognising student interaction patterns in exploratory learning environments. In *International Conference on Intelligent Tutoring Systems* (pp. 125–134). Springer.

Blessing, S., Gilbert, S., Ourada, S., & Ritter, S. (2007). Lowering the bar for creating model-tracing intelligent tutoring systems. *Frontiers in Artificial Intelligence and Applications, 158*, 443.

Boulden, D. C., Rachmatullah, A., Oliver, K. M., & Wiebe, E. (2021). Measuring in-service teacher self-efficacy for teaching computational thinking: Development and validation of the T-STEM CT. *Education and Information Technologies, 26*(4), 4663–4689. https://doi.org/10.1007/s10639-021-10487-2

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning, 23*(2), 170–185. https://doi.org/10.1080/10986065.2020.1779012

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Associatio*n, Vancouver, Canada (Vol. 1, p. 25).

Brusilovsky, P. (2003). Developing adaptive educational hypermedia systems: From design models to authoring tools. In *Authoring tools for advanced technology learning environments* (pp. 377–409). Springer.

Bunt, A., Conati, C., & Muldner, K. (2004). Scaffolding self-explanation to improve learning in exploratory learning environments. In *International Conference on Intelligent Tutoring Systems* (pp. 656–667). Springer.

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202-215.

Chen, M. (1995). A methodology for characterising computer-based learning environments. *Instructional Science, 23*(1–3), 183–220.

Clements, D. H., & Meredith, J. S. (1993). Research on Logo: Effects and efficacy. *Journal of Computing in Childhood Education, 4*(4), 263-290.

Clements, D. H., Battista, M. T., & Sarama, J. (2001). Logo and geometry. *Journal for research in mathematics education. Monograph, 10*, i-177. https://doi.org/10.2307/749924

Cocea, M., & Magoulas, G. (2009a). Context-dependent personalised feedback prioritisation in exploratory learning for mathematical generalisation. In *International Conference on User Modeling, Adaptation, and Personalization* (pp. 271–282). Springer.

Cocea, M., & Magoulas, G. D. (2009b). Hybrid model for learner modeling and feedback prioritisation in exploratory learning. *International Journal of Hybrid Intelligent Systems, 6*(4), 211–230.

Cocea, M., & Magoulas, G. D. (2010). Group formation for collaboration in exploratory learning using group technology techniques. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems* (pp. 103–113). Springer.

Cocea, M., Gutierrez-Santos, S., & Magoulas, G. (2008). Challenges for intelligent support in exploratory learning: The case of shapebuilder. In *Ist International Workshop on Intelligent Support for Exploratory Environments*. CEUR Workshop Proceedings.

Cui, Z., & Ng, O.-L. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research, 59*(5), 988–1012. https://doi.org/10.1177/0735633120979930

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33-39. https://doi.org/10.1145/2998438

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

Gal, Y., Yamangil, E., Shieber, S. M., Rubin, A., & Grosz, B. J. (2008). Towards collaborative intelligent tutors: Automated recognition of users' strategies. In *International Conference on Intelligent Tutoring Systems* (pp. 162–172). Springer.

Geraniou, E., Mavrikis, M., Hoyles, C., & Noss, R. (2010). A learning environment to support mathematical generalisation in the classroom. Proceedings of the *Sixth Congress of the European Society for Research in Mathematics Education*, 978-2-7342-1190-7. hal-02182374. https://hal.science/hal-02182374v1

Ginon, B., Jean-Daubias, S., Lefevre, M., Champin, P.-A., et al. (2014). Adding epiphytic assistance systems in learning applications using the sepia system. In *European Conference on Technology Enhanced Learning* (pp. 138–151). Springer.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Gutierrez-Santos, S., Mavrikis, M., & Magoulas, G. D. (2012). A separation of concerns for engineering intelligent support for exploratory learning environments. *Journal of Research and Practice in Information Technology, 44*(3), 347.

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends, 62*(6), 574–584. https://doi.org/10.1007/s11528-018-0290-9

Kaufmann, O. T. & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology,* 52(7), 1029-1048.

Karkalas, S., & Mavrikis, M. (2016a). Feedback authoring for exploratory learning objects: Authelo. In CSEDU 2016 – Proceedings of the *8th International Conference on Computer Supported Education* (Vol. 1, pp. 144–153). Science and Technology Publications, Lda.

Karkalas, S., Mavrikis, M., Xenos, M., & Kynigos, C. (2016b). Feedback authoring for exploratory activities: The case of a logo-based 3D microworld. In *International Conference on Computer Supported Education* (pp. 259–278). Springer.

Kastner-Hauler, O., Tengler, K., Sabitzer, B., & Lavicza, Z. (2025). A learning environment to promote the computational thinker: A Bebras perspective evaluation. In Z. Pluhár & B. Gaál (Eds.), *Informatics in schools. Innovative approaches to computer science teaching and learning* (Vol. 15228, pp. 85–98). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-73474-8_7

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75–86.

Klahr, D., & Nigam, M. (2004). The equivalence of learning paths in early science instruction: Effects of direct instruction and discovery learning. *Psychological Science, 15*(10), 661–667.

Kynigos, C. & Grizioti, M. (2018). Programming Approaches to Computational Thinking: Integrating Turtle Geometry, Dynamic Manipulation and 3D Space. *Informatics in Education, 17*(2), 321–340. https://doi.org/10.15388/infedu.2018.17

Kynigos, C. (2007). Half-baked logo microworlds as boundary objects in integrated design. *Informatics in Education-An International Journal, 6*(2), 335-359. https://doi.org/10.1007/s10758-007-9114-2

Kynigos, C., & Karavakou, M. (2023). Coding dancing figural animations: mathematical meaning-making through transitions within and beyond a digital resource. *Digital Experiences in Mathematics Education, 9*(2), 283-314. https://doi.org/10.1007/s40751-022-00118-x

Lee, H.-Y., Wu, T.-T., Lin, C.-J., Wang, W.-S., & Huang, Y.-M. (2024). Integrating computational thinking into scaffolding learning: An innovative approach to enhance science, technology, engineering, and mathematics hands-on learning. *Journal of Educational Computing Research, 62*(2), 431–467. https://doi.org/10.1177/07356331231211916

Mavrikis, M., Geraniou, E., Noss, R., & Hoyles, C. (2008). Revisiting pedagogic strategies for supporting students' learning in mathematical microworlds. In *Proceedings of the International Workshop on Intelligent Support for Exploratory Environments at EC-TEL* (Vol. 8, pp. 41-50).

Mavrikis, M., Geraniou, E., Gutierrez Santos, S. and Poulovassilis, A. (2019), Intelligent analysis and data visualisation for teacher assistance tools: The case of exploratory learning. *British Journal of Educational Technology*, 50: 2920-2942. https://doi.org/10.1111/bjet.12876

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*(1), 14.

Mitchelmore, M. C., & White, P. (2000). Development of angle concepts by progressive abstraction and generalisation. *Educational Studies in Mathematics, 41*, 209-238. https://doi.org/10.1023/A:1003927811079

Morales Gamboa, R., & Gamboa, R. M. (2000). Exploring participative learner modelling and its effects on learner behaviour.

Mumcu, F., Kıdıman, E., & Özdinç, F. (2023a). Integrating computational thinking into mathematics education through an unplugged computer science activity. *Journal of Pedagogical Research, 7*(2), 72-92. https://doi.org/10.33902/JPR.202318528

Mumcu, F., Uslu, N. A., & Yıldız, B. (2023b). Teacher development in integrated STEM education: Design of lesson plans through the lens of computational thinking. *Education and Information Technologies, 28*, 3443–3474. https://doi.org/10.1007/s10639-022-11342-8

Mumcu, F., Andic, B., Maricic, M., Tejera, M., & Lavicza, Z. (2025). Unpacking teachers' value beliefs about computational thinking and programming. *Journal of Educational Technology & Online Learning, 8*(1), 41-63. https://doi.org/10.31681/jetol.1497284

Ng, O.-L., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM – Mathematics Education, 53*(4), 847–860. https://doi.org/10.1007/s11858-020-01200-7

Noss, R., & Hoyles, C. (1996). Windows on mathematical meanings: *Learning cultures and computers (Vol. 17)*. Springer Science & Business Media.

Papert, S. (1980). *Computers for children. Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Pawar, U. S., Pal, J., & Toyama, K. (2006). Multiple mice for computers in education in developing countries. In *2006 International Conference on Information and Communication Technologies and Development* (pp. 64–71). IEEE.

Pearce-Lazard, D., Poulovassilis, A., & Geraniou, E. (2010). The design of teacher assistance tools in an exploratory learning environment for mathematics generalisation. In *European Conference on Technology Enhanced Learning* (pp. 260–275). Springer.

Polya, G. 1945. *How to Solve It: A New Aspect of Mathematical Method*. Princeton University Press: Princeton, New Jersey.

Refvik, K. A. S., & Bjerke, A. H. (2022). Computational thinking as a tool in primary and secondary mathematical problem solving: A literature review. *NOMAD Nordic Studies in Mathematics Education, 27*(3), Article 3. https://doi.org/10.7146/nomad.v27i3.149191

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: Implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272–283. https://doi.org/10.1080/10494820.2019.1612445

Turgut, M., Kohanová, I., & Gjøvik, Ø. (2024). Developing survey-based measures of mathematics teachers' pedagogical technology knowledge: A focus on computational thinking and programming tools. *Research in Mathematics Education*, 1–24. https://doi.org/10.1080/14794802.2024.2401482

Van Joolingen, W. R., & Zacharia, Z. C. (2009). Developments in inquiry learning. In *Technology-enhanced learning* (pp. 21–37). Springer.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

Weintrop, D., Morehouse, S., & Subramaniam, M. (2021). Assessing computational thinking in libraries. Computer Science Education, 31(2), 290–311. https://doi.org/10.1080/08993408.2021.1874229

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

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. (2011). Introducing computational thinking in education courses. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 465–470). https://doi.org/10.1145/1953163.1953297

Appendix: Interview Questions

- What specific characteristics do you believe make feedback effective for students?
- What is your perception about integrating computational thinking components into feedback?
- In your experience, how did feedback influence the development of your CT skills?
- What changes have you observed in your approach to problem-solving after receiving feedback?
- How do you perceive the relationship between programming, math education, and CT?
- How can educators better support students in developing computational thinking skills in math education through programming?