12ο Πανελλήνιο Συνέδριο με Διεθνή Συμμετοχή
"Διδακτική της Πληροφορικής"
Τόμος Πρακτικών
Επιμέλεια
Φεσάκης, Γ., Δημητρακοπούλου, Α., Σοφός, Α., Φωκίδης, Ε., & Κώστας Α.
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΕΤΠΕ

Enhancing Debugging Skills in Robotics Education: Effects of the MuSRA Model

Alexandra Papamargariti, Angeliki Dimitracopoulou

doi: [10.12681/cetpe.9218](https://doi.org/10.12681/cetpe.9218)

Enhancing Debugging Skills in Robotics Education: Effects of the MuSRA Model

Alexandra Papamargariti, Angeliki Dimitracopoulou
psed17008@aegean.gr, adimitr@aegean.gr
Laboratory of Learning Technology and Educational Engineering, Department of Preschool Education Sciences and Educational Design, University of the Aegean

Abstract

This study explores the use of the Multidimensional Support Model for Robotics Learning Activities (MuSRA) to help novice programmers develop debugging skills. MuSRA combines instructional strategies with digital tools to support metacognition. Central to the model is student-recorded digital data—screenshots, recordings, video captures—collected during robotics tasks. It positions students as reflective "scientists" analyzing their programming processes. The study involved 48 lower secondary school students with minimal programming experience in MuSRA-guided activities. The research examined (a) programming errors and (b) students' ability to correct them. Results showed common novice mistakes and improvements in debugging. Crucially, digital data reflection supported error detection. These findings suggest MuSRA is an effective approach to teaching debugging in educational robotics.

Keywords: debugging, digital tools, educational robotics activities, learning support, programming errors

Introduction

One of the key challenges novice programmers faces is identifying and correcting code errors. Debugging—a core programming process—is a complex cognitive skill. It fosters reflective thinking by prompting learners to evaluate and refine their reasoning (Papert, 1980). Research emphasizes debugging's role in building Computer Science (CS) competencies (McCauley et al., 2008), problem-solving skills (Jonassen, 2000), and computational thinking (Brennan & Resnick, 2012).

Effective debugging requires understanding programming concepts, mental models, and error patterns (Ducassé & Emde, 1989; Fitzgerald et al., 2008; Klahr et al., 1988). However, many novices lack these foundations (Perkins & Martin, 1986), making debugging difficult and frustrating (Alqadi & Maletic, 2017; Murphy et al., 2008).

Although vital, debugging is underemphasized in curricula; novices often rely on trial-and-error, highlighting the need for explicit strategies and support (McCauley et al., 2008; O'Dell, 2017). While instructors resolve errors quickly, learners need explicit strategies and support to develop independent debugging skills. The MuSRA, presented briefly in the present paper, addresses a gap in the field by offering a scalable and adaptable instructional model that supports structured yet flexible debugging for diverse learner needs. Its dual-layered design integrates predefined strategies and tools with customizable activity sheets, allowing teacher adaptation and student autonomy. This balance of structure and flexibility makes MuSRA suitable for various educational robotics contexts and responsive to different classroom dynamics.

Instructional Approaches and ER for Developing Debugging Skills

Recent studies have explored pedagogical interventions to improve debugging for novices. Chiu and Huang (2015) used game-based programs with embedded bugs and scaffolded worksheets to develop debugging strategies. DeLiema et al. (2019) showed that educator-led discussions at failure points enhance learning. Fields et al. (2021) had students create buggy projects, boosting motivation and error-recognition.

Educational robotics (ER) has emerged as a promising context for programming instruction, particularly in block-based platforms like Scratch and LEGO Mindstorms. The iterative, trial-and-error nature of robotics tasks naturally integrates debugging, providing an ideal environment for novices (Eguchi & Uribe, 2017; Scaradozzi et al., 2019). However, the effectiveness of ER in enhancing debugging skills remains inconclusive. Research has shown that overly structured or pre-designed robotics activities can limit learners' engagement with debugging, reducing them to passive participants and hindering the development of critical thinking and problem-solving skills (Xia & Zhong, 2018).

Few ER frameworks offer adaptable support for diverse learner needs. For instance, Eguchi and Uribe (2017) designed a learner-centered ER unit with 10 science-integrated robotics tasks, using checklists, journals, and peer support to foster debugging. Atmatzidou and Dimitriadis (2017) proposed the CPG+ model, emphasizing problem-solving and collaboration through 12 structured activities; guided instruction was linked to improved student performance (Atmatzidou et al., 2018). Chevalier et al. (2020, 2022) introduced the CCPS model, promoting recursive debugging via iterative design and delayed feedback. Socratous and Ioannou (2021) found that structured ER tasks enhanced debugging, while unstructured ones boosted engagement.

Despite the progress made with various educational robotics frameworks, a gap remains in providing scalable, adaptable instructional models that integrate robust debugging support in ER environments. The field continues to explore ways to balance structure and flexibility to better address the diverse learning needs of students, ultimately improving debugging performance and fostering deeper engagement in programming tasks (Blancas et al., 2020; Daniela, 2019).

The present study

To address existing gaps, this study introduces the Multidimensional Support Model for Robotics Learning Activities (MuSRA), a structured framework designed to enhance students' programming and debugging skills in educational robotics (ER) contexts. MuSRA integrates targeted instructional strategies with digital tools to foster the development of these skills. A central element of the model is the use of "student-recorded digital data" (e.g., screenshots, screen recordings, and video captures) to guide students throughout the programming process. Recent research findings revealed positive students' perceptions of the implementation of MuSRA, in terms of suitability of learning activities, effectiveness and user-friendliness of incorporated digital activity sheets, while their dual-modality approach—offering both written and oral response options—enhance inclusivity, accommodating diverse learning needs (Papamargariti & Dimitracopoulou, 2025).

This study examines MuSRA's effectiveness in a real-world educational robotics setting by analyzing error types, correction strategies and the overall improvement in their debugging performance. By embedding reflective practices and strategic debugging support into robotics activities, this research aims to advance programming education, providing new insights into effective approaches for teaching debugging to novice learners in interactive, hands-on learning environments.

Multidimensional Support Model for Robotics Learning Activities [MuSRA]

The MuSRA model is an instructional framework designed to enhance ER learning by combining structured pedagogical strategies with integrated digital tools. It offers teachers the capacity to adapt supports to various scenarios while ensuring consistent outcomes. Teachers can tailor the digital activity sheets to curriculum goals, student profiles, and instructional styles, while students retain agency through multimodal input tools and differentiated guidance. This balance of structure and autonomy makes MuSRA scalable across varied ER settings. MuSRA is built around four key components: (I) the actors (students and teacher), (II) the robotics learning activities, (III) the technological and digital tools, and (IV) the MuSRA digital activity sheets, along with their integrated strategies and internal digital learning support tools.

Actors: In MuSRA, students act as scientists, documenting their learning through screenshots, robot execution videos, and experiment recordings. This fosters metacognitive and problem-solving skills, autonomy, and technological fluency. Digital activity sheets support diverse learning styles through three response modes: text typing, voice typing, and audio recording, accommodating diverse learning needs. MuSRA also supports teachers by reducing workload and improving classroom orchestration (Dillenbourg, 2013). It enables real-time monitoring of student progress, helps manage instructional time, and identifies key intervention moments for discussions or feedback.

Robotics Learning Activities: Based on Komis et al. (2017), MuSRA's robotics learning activities are student-centered, co-creative, and grounded in real-world problems. Two activity types are used: Robotics Tasks and a Final Challenge. Robotics Tasks follow Pólya's (1945) four-phase model to progressively develop problem-solving skills.

Technological and Digital Tools: MuSRA combines two student workspaces: (1) The Robotics Environment (RE) with a robotics kit (e.g., LEGO EV3 or Spike Prime) and a programming tablet, and (2) The Digital Activity Sheet Environment (DASE), featuring a second tablet with the Digital Activity Sheet Platform (DASP) and apps like camera, audio recorder, and notepad. This setup enables students to integrate multimodal inputs—screenshots, videos, reflections. The teacher's workspace includes a centralized DASP to organize submissions and monitor progress.

MuSRA Digital Activity sheets: The MuSRA model centers on Digital Robotics Activity Sheets (mRASs), which guide students' learning across five key dimensions: (1) Phase-Based Guidance: mRASs follow Pólya's four problem-solving phases and add a fifth "Discussion" phase to encourage reflection and iterative improvement. (2) Tools Management: Students receive clear instructions for using tools (e.g., screenshots, videos, note-taking) to smoothly transition between tasks and manage resources effectively. (3) Collaboration Support: Role-based teamwork enhances cooperation. Roles rotation include: analyst (manages DASE and records responses), programmer (builds and runs code), and debugger (detects errors and tracks progress). (4) Cognitive Support: Acting as scientists, students collect and analyze their own digital data (e.g., program screenshots, video recordings) to support observation and self-regulated learning. (5) Metacognitive Support: Three embedded strategies promote reflection and self-regulation: (a) Self-Explanation: Students clarify their thinking and use visuals like flowcharts. (b) Debugging via Digital Record Analysis: Students analyze uploaded code screenshots and videos to improve solutions. (c) Metacognition Monitoring: Students reflect on their performance, challenges, and learning strategies at the end of tasks.

To enhance learning and reflection, MuSRA integrates three Digital Internal Tools (DiTs) within the mRASs: (1) Recordings Tool: lets students upload screenshots and videos to track and refine work. (2) Verbal Expression Tool: offers text, voice typing, or audio responses, promoting accessibility. (3) Keep the Important Tool: compiles a multimedia notebook with key insights, visuals, and audio notes, fostering reflection and knowledge consolidation.

Figure 1 illustrates how each DiT maps onto specific phases of the problem-solving model, supporting cognitive and metacognitive engagement throughout the activity. MuSRA is most effectively applied in lower secondary educational settings where students are beginning to learn programming within structured, classroom-based robotics activities. Its integration of digital supports and multimodal data collection is particularly suited to block-based programming platforms (e.g., Scratch, LEGO EV3), where learners benefit from guided debugging strategies. MuSRA is also well aligned with curricula emphasizing collaborative problem solving, self-regulation, and inquiry-based STEM learning, making it a robust framework for formal education environments with access to basic robotics kits and tablet-based digital tools.

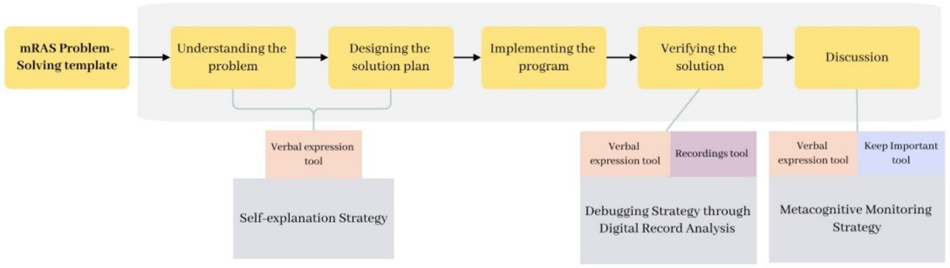
mRAS Problem-Solving template
Understanding the problem
Designing the solution plan
Implementing the program
Verifying the solution
Discussion
Verbal expression tool
Self-explanation Strategy
Verbal expression tool
Recordings tool
Debugging Strategy through Digital Record Analysis
Verbal expression tool
Keep Important tool
Metacognitive Monitoring Strategy

Figure 1. MuSRA Strategies & Digital Internal Tools (DiTs)

Methodology

This study aims to evaluate the effectiveness of the MuSRA model in supporting novice programmers in developing effective debugging skills, during robotics programming activities. To achieve this goal, it examines the impact of MuSRA on students' debugging performance, focusing on the types and frequency of programming errors, as well as students' ability to identify and resolve these errors over time. To address the research objectives, the study is guided by the following research questions (RQ):

1. What types and how often did programming errors occur during robotics activities?
2. What is the impact of the MuSRA model on students' programming errors over time?
3. How did students perceive the use of self-recorded digital data for debugging?
4. What is the impact of the MuSRA model on students' overall debugging performance?

Regarding participants, the study involved 48 lower secondary school students, evenly split by gender (24 girls and 24 boys). They were organized into four robotics club groups, each consisting of 12 students and utilizing either Lego Mindstorms EV3 or Lego Spike Prime kits. Within each group, students worked in teams of three and rotated through the roles of analyst, programmer, and debugger during the activities. Prior to participation, students completed a background questionnaire ensuring diversity in gender, grade, and experience. Most participants were beginners, with 44 students reporting minimal or no prior programming experience, and 68.8% (33 students) indicating they had never used a robotics kit before.

A mixed-methods approach (combining quantitative data with qualitative insights) was employed to analyze the frequency and nature of programming errors, as well as students' debugging performance within the MuSRA implementation. The research tools include: (i) Debugging Test: Conducted at two points (mid-debugging and post-debugging tests), this assessment evaluated students' ability to identify and correct errors in predefined tasks. Each test was scored on error identification (5 points) and proposed solutions (5 points), for a total of 10 points per item. Cronbach's alpha (0.887) indicated strong internal consistency. (ii) Survey Feedback Questionnaire: Administered after the final challenge, this survey captured students' perceptions of the MuSRA model and the role of self-recorded digital data in supporting debugging strategies, focusing on error identification, resolution, and understanding. (iii) Focus Group Interviews: Semi-structured interviews (50-60 minutes) conducted one week after the intervention provided students with the opportunity to reflect on their debugging and problem-solving development. Content analysis of transcriptions identified key themes. (iv) Observation: Structured classroom observations recorded real-time error identification, correction attempts, and debugging strategies.

The research procedure, illustrated in Figure 2, involved a nine-week intervention during which each robotics club class took part in nine three-hour sessions (Papamargariti & Dimitracopoulou, in press).

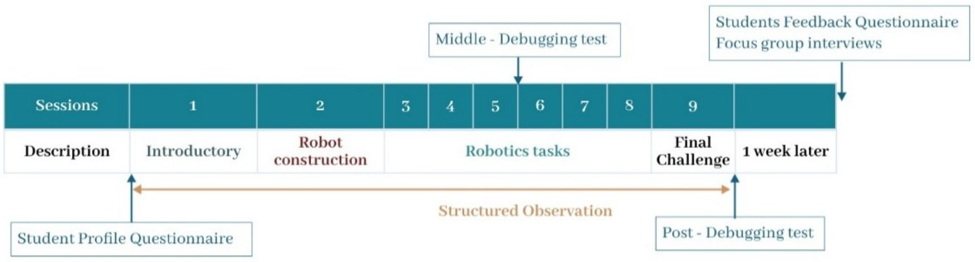

Figure 2. Research Procedure

The intervention began with an introductory session in which students were introduced to robotics kits and programming environments. In the following construction session, students worked in groups to build robot vehicles. They then progressed through six robotics problem-solving tasks of increasing complexity. After session 5, students completed a mid-debugging test. Upon finishing all tasks, they took the post-test for debugging. The final challenge session required students to solve a complex robotics problem according to specific evaluation criteria. One week later, students completed a feedback survey and semi-structured interviews (Figure 2).

Research results

Types and frequency of programming errors

To address RQ1, quantitative data were gathered across all sessions. Researchers categorized student programming errors into seven types using completed activity sheets and structured observation logs. These data sources captured code iterations, debugging attempts, and real-time behaviors, enabling thorough analysis of error frequency and patterns throughout the intervention.

As shown in Table 1, the most frequent error (22.44%) was assigning incorrect values to block variables, such as improper distances or speeds, which often led to unexpected robot behavior. The second most common error (16.67%) involved omitting essential programming blocks—like loops or control structures—resulting in incomplete or non-functional programs. Variable confusion (14.74%), such as mixing motor power with distance, was also prevalent. Incorrect port assignments (14.10%) highlighted difficulties in hardware-software integration. Errors in conditional logic and sequencing (both 12.82%) reflected challenges in understanding logical flow. Finally, non-programming issues (6.41%), including improper robot placement or assembly mistakes, pointed to limited awareness of the physical programming context.

The findings highlight persistent challenges in variable management, logical sequencing, and hardware integration, emphasizing the need for instructional approaches that support iterative testing, reflective practice, and systematic debugging. The identified error types align with prior research. Three common errors in this study—defining values in block variables, selecting block sequences, and setting conditions—were also found in non-robotic, non-block-based programming studies (Chiu & Huang, 2015; Liu et al., 2017). Kim et al. (2018) reported similar challenges in variable assignment and block arrangement among novice robotics learners. Similarly, Socratous and Ioannou (2020; 2021) observed recurring issues with variable configuration, sequencing, and logic, highlighting persistent difficulties in block-based programming across educational contexts.

Table 1. Types and frequency of programming errors observed during robotics tasks

Error Type	Description	Fr/ncy	Fr/ncy %	Supported by Prior Studies
Error in defining a value to a block's variable	Not accurate or wrong calculation of the value of a variable	35	22.44%	Kim et al., 2018; Socratous & Ioannou, 2021; Chiu & Huang, 2015; Liu et al., 2017
Error selecting the appropriate variable in a block	Choosing the wrong variable within the same block	23	14.74%	Kim et al., 2018; Socratous & Ioannou, 2020
Error selecting the correct block or the correct sequence of blocks	Use of an inappropriate block	20	12.82%	Socratous & Ioannou, 2021; Chiu & Huang, 2015; Liu et al., 2017
Error in matching a motor or a sensor to the correct port	Motors connected to the wrong ports	22	14.10%	Kim et al., 2018; Socratous & Ioannou, 2020
Error in defining conditions	Difficulty understanding conditional logic	20	12.82%	Kim et al., 2018; Socratous & Ioannou, 2021; Chiu & Huang, 2015; Liu et al., 2017
Error recognizing external factors as the cause of the program's failure	Failure to recognize a program without errors	10	6.41%	—
Error due to a missing block	Missing essential blocks	26	16.67%	Kim et al., 2018; Socratous & Ioannou, 2020

Impact of the MuSRA model on programming errors

To address RQ2, a sequential analysis of errors across seven tasks was conducted at the team level using activity sheets (tracking trials and debugging reflections) and structured observation sheets (capturing real-time mistakes). As summarized in Table 2, a decline in errors suggests the MuSRA model's potential effectiveness in supporting learning and progressive error correction.

Table 2. Number of programming errors across robotics tasks

Challenge	Robotic Task	Number of Errors	Number of Errors %
1	Labyrinth	32	20,51%
2	Follow the colors	28	17,95%
3	Make the right choice and avoid the wall	26	16,67%
4	Move a Lego block	23	14,74%
5	Move a box to the storage position	20	12,82%
6	Move two boxes to the collection areas	16	10,26%
7	Final Challenge	11	7,05%

In the initial tasks—Labyrinth (32 errors; 20.51%) and Follow the Colors (28 errors; 17.95%)—error frequencies were highest, reflecting students' early exposure to robotics, limited programming knowledge, and difficulty understanding task requirements. As students progressed, a decline in errors was observed. In intermediate tasks such as Make the Right Choice and Avoid the Wall (26 errors; 16.67%), Move the Cube (23 errors; 14.74%), and Move the Box to the Storage Position (20 errors; 12.82%), students showed improvement to apply learned concepts. This trend continued in advanced tasks—Move Two Boxes to the Collection Areas (16 errors; 10.26%) and the Final Challenge (11 errors; 7.05%)—indicating sustained learning gains. The steady reduction in errors suggests the MuSRA model effectively promotes skill acquisition, enhances debugging, and supports improved performance in increasingly complex robotics programming challenges.

Students' perceptions of the use of self-recorded digital data

To address RQ3, data were collected from a Survey Feedback Questionnaire and semi-structured focus group interviews after the final challenge. The survey focused on students' perceptions of the MuSRA model and the role of student-recorded digital data in debugging. Table 3 shows that 83% of students agreed that recording and reviewing data helped them recognize errors more effectively, enhancing reflection and awareness. Additionally, 79% felt that reviewing data improved their error-solving ability, suggests self-recording aided troubleshooting.

Table 3. Perceived effectiveness of student-recorded digital data

Category	Disagree (%)	Neutral Opinion (%)	Agree (%)
Error Recognition	4%	13%	83%
Error-Solving Ability	0%	21%	79%
Problem-Solving Strategies	3%	19%	78%
Debugging Strategies	3%	20%	77%

Additionally, 78% of students reported that digital data helped develop their problem-solving strategies, suggesting reflection refined their approach to complex tasks. Furthermore, 77% felt it improved their debugging skills, indicating its role in fostering strategic problem-solving. Qualitative data from focus group interviews highlighted the benefits of recording the debugging process. One student shared, "When I had to correct a programming error, recording my efforts helped me see my mistakes and fix them." Overall, both survey and interview findings underscore the significant impact of student-recorded digital data on debugging, self-regulated learning, and programming proficiency.

Students' debugging performance

To address RQ4, a comparison of students' results on the Mid-Debugging Test and Post-Debugging Test revealed significant improvements in debugging skills. Statistical analysis using a paired samples t-test (Table 4) showed an increase in average scores from 65.00 (*SD* = 20.82) on the mid-test to 76.91 (*SD* = 16.89) on the post-test, with a mean improvement of 11.92 points, *t*(47) = -7.83, p = 0.00. Specifically, students improved in error identification, with scores rising from 67.97 to 79.99, *t*(47) = -6.55, *p* = 0.00, and in suggesting corrections, from 62.64 to 75.33, *t*(47) = -7.12, *p* = 0.00. These findings suggest that the MuSRA model effectively enhanced students' debugging abilities, particularly in error detection. The greater improvement in identification over correction emphasizes the ongoing challenge students face in implementing fixes.

Table 4. Debugging Performance

	Mid-Test		**Post-Test**		**Paired Differences**	***t***	**Paired Samples *t*-Test**
	Mean	***SD***	***Mean***	***SD***	***Mean***		
Debugging Test Scores	65.00	20.82	76.91	16.89	-11.92	-7.83	0.00*
Found the Error	67.97	21.18	79.99	15.18	-12.02	-6.55	0.00*
Proposed a Solution	62.64	21.70	75.33	16.22	-12.69	-7.12	0.00*

Note. *$p < 0.05$

Discussion

The study provides strong evidence for the effectiveness of the MuSRA model in overcoming common programming challenges in robotics. By analyzing errors, student perceptions, and debugging performance, it offers key insights that both align with and extend existing research on supporting novice learners in robotics-based activities.

Regarding RQ1 (Types and frequency of programming errors), the most common errors included incorrect variable values, missing blocks, and incorrect block or port selections. Sequencing errors and misuse of conditional logic revealed struggles with logical flow and decision-making. These findings align with prior research, such as Kim et al. (2018), who observed similar issues in variable definition and block placement in novice robotics learners. Socratous and Ioannou (2020; 2021) also identified similar challenges in variable configuration and sequencing, highlighting the persistence of these issues.

Regarding RQ2 (Impact on Programming Error Reduction), data showed a consistent decline in errors across tasks, indicating that the MuSRA model effectively supported learning. As students advanced, their programming and debugging skills improved, highlighting scaffolding and the value of iterative learning and reflection.

Regarding RQ3 (Perceived Effectiveness of Student-Recorded Digital Data), survey results showed that 83% of students believed that recording and reviewing digital data improved error recognition, problem-solving, and debugging strategies. This finding aligns with Atmatzidou and Dimitriadis (2017), who emphasized the role of tracking progress in fostering a deeper understanding of computational concepts. The reflective process, enabled by student-recorded digital data, helped students reflect and self-regulate and foster autonomy in problem-solving.

Regarding RQ4 (Students' Debugging Performance), the MuSRA model led to significant improvements in students' debugging skills, particularly in error identification and corrective actions. This aligns with Chevalier et al. (2020), who found that structured interventions in educational robotics enhance troubleshooting skills. However, while error detection improved, correcting errors showed less progress, supporting Fitzgerald et al. (2008), who highlighted the greater difficulty novices face in implementing solutions.

Conclusion

Overall, the MuSRA model effectively supported novice learners in developing programming and debugging skills, addressing issues like variable misconfiguration, sequencing errors, and logic misuse. Through digital self-recording, it promoted engagement that improved performance. The findings highlight the value of scaffolded, reflective learning in educational robotics.

This study contributes to programming education by evaluating MuSRA—a model that integrates iterative learning and self-recorded data to address debugging challenges. It offers insights into common novice errors, aligns with existing research, and emphasizes the need for structured support. The benefits of student-recorded data underscore reflection's role in building autonomy and metacognitive awareness.

While this study offers insights into effectiveness of the MuSRA model, it is limited by the use of a small, context-specific sample, which may restrict the generalizability of the findings. Additionally, the research focused on short-term outcomes. Future research could explore how MuSRA supports students with different learner profiles or educational levels to identify practices for tailoring the model across diverse contexts.

References

Alqadi, B. S., & Maletic, J. I. (2017, March). An empirical study of debugging patterns among novices programmers. *Proceedings of the ACM SIGCSE technical symposium on computer science education* (pp. 15-20). ACM.

Atmatzidou, S., & Demetriadis, S. (2017). A didactical model for educational robotics activities: A study on improving skills through strong or minimal guidance. In D. Alimisis, M. Moro, & E. Menegatti (Eds.), *Proceedings of the Educational Robotics in the Makers Era. Edurobotics 2016 Conference* (pp. 58-72). Springer.

Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How does the degree of guidance support students' metacognitive and problem solving skills in educational robotics? *Journal of Science Education and Technology, 27*, 70-85.

Blancas, M., Valero, C., Mura, A., Vouloutsi, V., & Verschure, P. F. (2020). "CREA": An inquiry-based methodology to teach robotics to children. In M. Merdan, W. Lepuschitz, G. Koppensteiner, & D. Obdržálek (Eds.), *Robotics in Education: Current research and innovations 10* (45-51). Springer.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 AERA Annual meeting* (Vol. 1, p. 25). AERA.

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: A model for creative computational problem solving. *International Journal of STEM Education*, 7, 1-18.

Chevalier, M., Giang, C., El-Hamamsy, L., Bonnet, E., Papaspyros, V., Pellet, J. P., Audrin, K, Romero M., Baumberger. B., & Mondada, F. (2022). The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. *Computers & Education, 180,* 104431.

Chiu, C. F., & Huang, H. Y. (2015). Guided debugging practices of game based programming for novice programmers. *International Journal of Information and Education Technology*, *5*(5), 343.

Daniela, L. (2019). *Smart learning with educational robotics.* Springer International Publishing.

DeLiema, D., Dahn, M., Flood, V. J., Asuncion, A., Abrahamson, D., Enyedy, N., & Steen, F. (2019). Debugging as a context for fostering reflection on critical thinking and emotion. In. E. Manalo (Ed.), *Deeper learning, dialogic learning, and critical thinking, research-based strategies for the classroom* (pp. 209-228). Routledge.

Dillenbourg, P. (2013). Design for classroom orchestration. *Computers & Education, 69*, 485-492.

Ducassé, M., & Emde, A. M. (1989). A review of automated debugging systems: Knowledge, strategies and techniques. *Proceedings of the 11th International Conference on Software Engineering* (pp. 162-163). IEEE

Eguchi, A., & Uribe, L. (2017). Robotics to promote STEM learning: Educational robotics unit for 4th grade science. *Proceedings of the 2017 IEEE Integrated STEM Education Conference (ISEC)* (pp. 186-194). IEEE.

Fields, D. A., Kafai, Y. B., Morales-Navarro, L., & Walker, J. T. (2021). Debugging by design: A constructionist approach to high school students' crafting and coding of electronic textiles as failure artefacts. *British Journal of Educational Technology, 52*(3), 1078-1092.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education, 18*(2), 93-116.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, *48*(4), 63-85.

Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C.N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers & Education*, *91,* 14-31.

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362-404.

Komis, V., Romero, M., & Misirli, A. (2017). A scenario-based approach for designing educational robotics activities for co-creative problem solving. In D. Alimisis, M. Moro & E. Menegatti (Eds.), *Educational robotics in the makers era 1* (pp. 158-169). Springer.

Li, C., Chan, E., Denny, P., Luxton-Reilly, A., & Tempero, E. (2019). Towards a framework for teaching debugging. *Proceedings of the 21st Australasian Computing Education Conference* (pp. 79-86). Association for Computing Machinery.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review from an educational perspective. *Computer Science Education*, *18*(2), 67-92.

Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: the good, the bad, and the quirky--a qualitative analysis of novices' strategies. *ACM SIGCSE Bulletin*, *40*(1), 163-167.

O'Dell, D. H. (2017). The Debugging Mindset: Understanding the psychology of learning strategies leads to effective problem-solving skills. *Queue, 15*(1), 71-90.

Papamargariti & Dimitracopoulou (2025). Multidimensional support model for robotics learning activities. [Manuscript submitted for publication].

Papert, S. (1980). *Mindstorms: Children, computers and powerful Ideas.* Perkins & Martin.

Pólya, G., & Szegö, G. (1945). Inequalities for the capacity of a condenser. *American Journal of Mathematics*, 67(1), 1-32.

Scaradozzi, D., Screpanti, L., Cesaretti, L. (2019). Towards a definition of educational robotics: A classification of tools, experiences and assessments. In L. Daniela (Ed.) *Smart learning with educational robotics* (pp. 63-92). Springer.

Socratous, C., & Ioannou, A. (2020). Common errors, successful debugging, and engagement during block-based programming using educational robotics in elementary education. *Proceedings of the 14th International Conference of the Learning Sciences* (pp. 991–998). International Society of the Learning Sciences.

Socratous, C., & Ioannou, A. (2021). Structured or unstructured educational robotics curriculum? A study of debugging in block-based programming. *Educational Technology Research and Development*, *69*(6), 3081-3100.

Xia, L., & Zhong, B. (2018). A systematic review on teaching and learning robotics content knowledge in K-12. *Computers & Education*, *127*, 267-282.