

Future Trends in Computer Programming Education

Stelios Xinogalos 

Department of Applied Informatics, School of Information Sciences, University of Macedonia,
GR-54634 Thessaloniki, Greece; stelios@uom.edu.gr

1. Introduction

Programming education has attracted the interest of instructors and researchers for several decades [1]. It is widely accepted that learning programming is difficult [2] and the classic approach to teaching programming [3], which is based on using a conventional programming language and environment for solving number and symbol processing problems, is quite problematic. Students must be supported, motivated and engaged in the learning process in order to comprehend the principles of programming and write good-quality code. Several types of educational programming environments and tools have been developed over the decades, such as programming microworlds and games, structure editors, program/software visualization tools, flowchart-based programming environments and so on [4,5]. Several teaching approaches have also been devised based on the aforementioned tools. Moreover, extended research has been carried out regarding the best strategy for introducing novices to programming, including the imperative-first, functions-first and objects-first strategies, as well as the most appropriate language for each strategy.

In recent years the educational community around the world has shown an even greater interest in teaching and learning programming as a means of promoting Computational Thinking (CT) [6], which is considered important for everyone nowadays, not just computer scientists. Primary school kids nowadays are educated to think computationally through specially designed programming environments, such as Scratch [7] or the widely known “Hour of Code” [8,9] games.

Another important dimension of the work around teaching and learning programming nowadays lies in the way that “digital natives” [10] grow up and learn. Game-based learning and educational/serious games are considered a promising approach for learning programming. Pair programming [11] and more recently distributed pair programming [12,13] is another contemporary approach that promotes collaboration among students, which is also considered important. Most recently, the advent of LLMs and generative AI tools has started transforming the computer programming education landscape [14].

The objective of the Special Issue (SI) on Future Trends in Computer Programming Education was to explore recent advances, as well as future trends in Computer Programming Education. This SI consists of eleven articles that were accepted in the SI after a rigorous peer-review process. Thirty-one researchers from China, Greece, Mexico, Portugal, Russia, Slovakia, Spain, Taiwan and Vietnam contributed their work in this SI.

The eleven articles included in the SI cover a wide range of issues related to computer programming education, as can be seen in the word cloud of the keywords included in them and presented in Figure 1. In the following paragraphs, the eleven articles are summarized, providing their goals, methodology, findings and implications for future research. The articles are organized in four thematic areas.



Received: 24 July 2026

Revised: 29 July 2026

Accepted: 1 August 2026

Published: 3 August 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

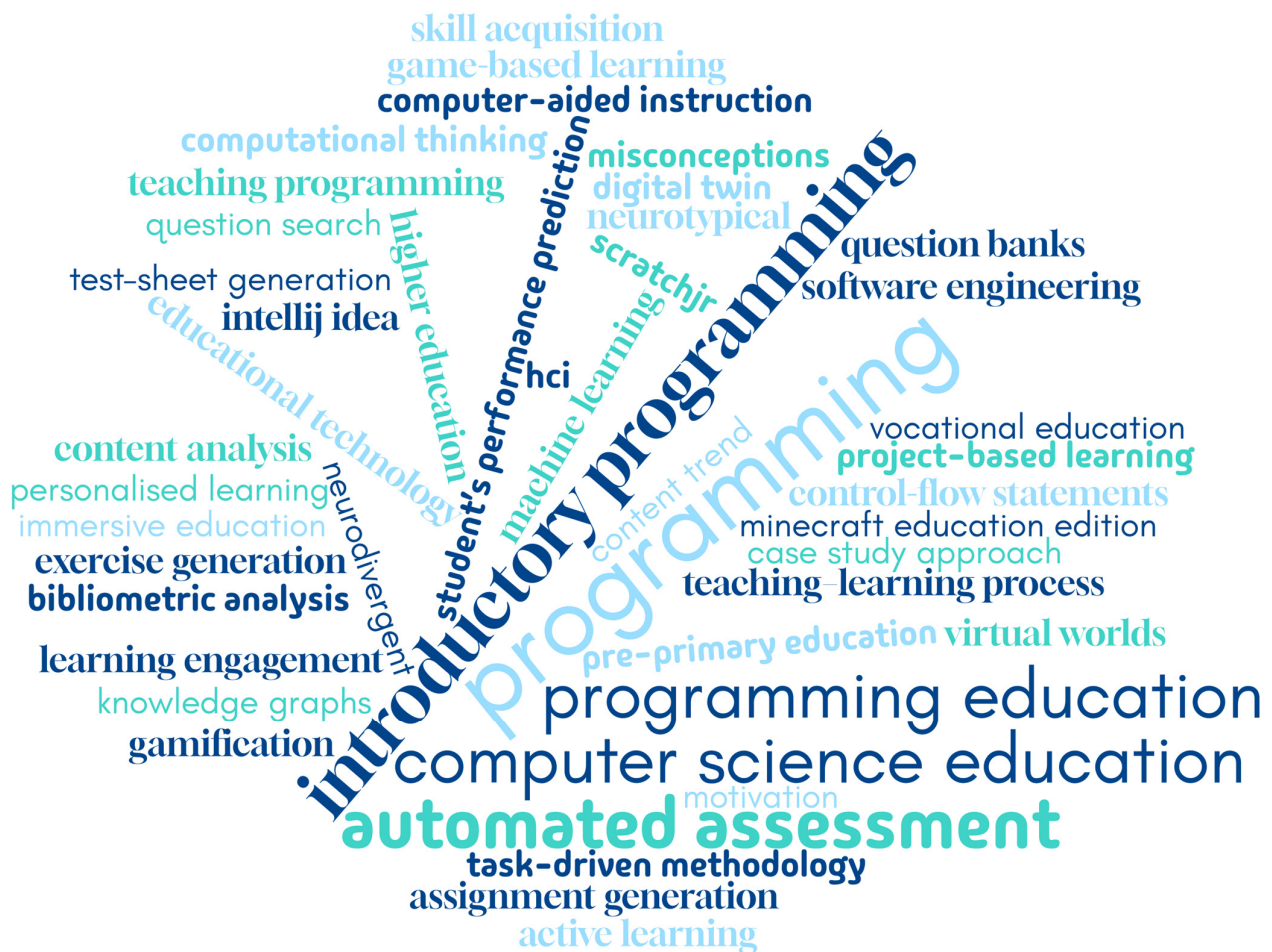


Figure 1. Word cloud of keywords included in the articles of the Special Issue.

2. Computational Thinking and the Computer Science Curriculum

The first article in this thematic area presents a bibliometric and content analysis of research on enhancing CT (contribution 1). This is followed by an article presenting a study on Computer Science teachers' views regarding the recent reform of the Computer Science curriculum in lower secondary education in Greece, which aims at cultivating CT, problem solving and digital skills, among others (contribution 2). The next article presents the design and evaluation of an agent-based training system aimed at helping both neurotypical students and students with special needs learn programming concepts with ScratchJr, an environment that can be used for introducing pre-primary students to CT (contribution 3).

The article "Study Trends and Core Content Trends of Research on Enhancing Computational Thinking: An Incorporated Bibliometric and Content Analysis Based on the Scopus Database" by Ling-Hsiu Chen and Ha Thi The Nguyen (contribution 1) provides a systematic review of research on CT over the period of 2008–2022. It employs a combined bibliometric and content analysis of 132 publications indexed in Scopus to identify publication patterns, influential contributors, and thematic developments in the field. The findings reveal a steady growth in CT-related research, with a notable increase in studies integrating game-based learning approaches. A peak in publication activity was recorded in 2021, while the analysis also highlights geographic trends, indicating that the United States leads in research output, while China has the highest citation impact. In terms of thematic content, the study identifies core research focusing on strategies, learner attitudes, and instructional methods aimed at enhancing CT skills, particularly through interactive and

technology-supported learning environments. Additionally, keyword and co-occurrence analyses demonstrate the centrality of educational applications in shaping the evolution of the field. The authors conclude that understanding these trends can inform future research directions, policymaking, and curriculum design, while the inclusion of other databases (like WOS and PubMed) and alternative analytical tools (like VOSviewer and CiteSpace) in subsequent studies can improve comprehensiveness.

The article “Enacting Computer Science Curriculum Reform: The Case of Model and Experimental Lower Secondary Schools in Greece” by Dimitrios Yiatas and Athanassios Jimoyiannis (contribution 2) investigates teachers’ views regarding the implementation of a new Computer Science curriculum in the Greek model and experimental lower secondary schools. The study is based on qualitative data (interviews), examining how educators interpret and apply the curriculum in real classroom settings. Findings show that teachers generally adopt the curriculum’s student-centered philosophy, using it flexibly as a guide rather than a rigid framework. The curriculum promotes key competencies such as CT, collaboration, creativity, and problem solving, aligned with modern educational goals. Teachers report using innovative teaching methods, such as projects, programming activities, and digital tools, as well as alternative assessment practices. Overall, the study concludes that successful implementation is supported by teacher expertise, autonomy, and a supportive school environment, highlighting the importance of flexibility in curriculum design.

The article “A Preliminary Usability Study of a Novel Educational Training System to Teach ScratchJr. in School” by Maria Jesus Manzanares, Diana Perez Marin and Celeste Pizarro (contribution 3) presents the design and pilot evaluation of an interactive agent-based training system aimed at helping young students learn programming with ScratchJr. It focuses on early childhood education, exploring how an agent-based interactive system can support neurotypical students and students with special needs in learning basic coding concepts. The study conducts a usability evaluation to assess how easy, engaging, and effective the system is for classroom use. Findings indicate that the system is generally user-friendly and enhances students’ engagement and understanding of fundamental programming ideas.

3. Misconceptions on Control-Flow Statements and Performance Prediction

The first study (contribution 4) in this thematic area investigates students’ misconceptions on control-flow statements with the aim of designing targeted instructional interventions and intelligent tutoring systems aimed at improving fundamental programming skills. The next article (contribution 5) utilizes datasets, including academic records, analyzed by machine learning algorithms and identified metrics for predicting students’ performance.

The article “Explain Trace: Misconceptions of Control-Flow Statements” by Oleg Sychev and Mikhail Denisov (contribution 4) investigates common misconceptions that novice programmers exhibit when reasoning about control-flow constructs. It adopts an empirical methodology based on a large set of 181 specially designed questions, each combining pseudocode with execution traces that may contain deliberate errors, prompting students to explain the correctness or incorrectness of specific steps. The study analyzes 10,799 student responses collected from 67 participants, enabling a systematic identification and classification of misconceptions related to sequences, loops, and conditional statements. Findings reveal 24 distinct misconceptions, of which some align with previously documented issues, while others represent novel or more fine-grained variants not extensively discussed in earlier literature. The authors highlight that misconceptions about control flow are often underexplored as a distinct category, despite their central role in introductory programming comprehension. They emphasize the value of open-ended explanatory tasks for uncovering

students' mental models, as opposed to relying solely on multiple-choice or correctness-based assessments. Overall, the article concludes that a more systematic understanding of these misconceptions can inform the design of targeted instructional interventions and intelligent tutoring systems aimed at improving fundamental programming skills.

The article "Predicting Student Performance in Introductory Programming Courses" by João P. J. Pires, Fernanda Brito Correia, Anabela Gomes, Ana Rosa Borges and Jorge Bernardino (contribution 5) reviews literature on the use of machine learning techniques to identify students at risk of poor performance in early programming education. It situates the problem within the broader challenge of high failure and dropout rates in introductory programming courses in Higher Education, emphasizing the need for early intervention mechanisms. Methodologically, the study evaluates the performance of various algorithms, using identified metrics and considering the applicability in Higher Education Introductory Programming courses. The results demonstrate that these models can achieve meaningful predictive accuracy, enabling instructors to detect at-risk students at an early stage. The findings highlight the potential of data-driven approaches to support targeted pedagogical interventions, thereby improving student success and retention. Overall, the study concludes that machine learning-based prediction constitutes a valuable tool in Computer Science Education, though its effectiveness depends on data quality and model selection.

4. Teaching Approaches and Tools for Enhancing Programming Education

The first article in this thematic area investigates students' engagement and skill acquisition in introductory programming courses utilizing digital-twin technology instead of traditional learning approaches (contribution 6). The next two articles propose two more alternatives to the traditional learning approach, namely task-driven case studies that combine the task-driven methodology with the case study approach (contribution 7) and gamified project-based learning focused on Vocational Education (contribution 8). The last paper in this thematic area refers to the use of Virtual Worlds in programming education for dealing with common difficulties that accompany the traditional learning approaches for programming (contribution 9).

The article "Digital Twin-Enhanced Programming Education: An Empirical Study on Learning Engagement and Skill Acquisition" by Ming Lu and Zhongyi Hu (contribution 6) explores the use of digital twin (DT) technology in introductory programming courses. It addresses limitations of traditional teaching, which often fails to connect theory with practice and results in low student engagement and weak problem-solving skills. Using a quasi-experimental study with 135 students, it compares a DT-based learning environment with conventional instruction. Results show that students using digital twins demonstrate higher engagement (behavioral, cognitive, emotional) and improved performance in coding tasks. This approach is especially beneficial for beginners, significantly enhancing algorithmic thinking, debugging, and practical programming skills. Overall, the study concludes that digital twin technology provides an immersive, interactive learning model that effectively bridges theory and practice in programming education.

The article "Teach Programming Using Task-Driven Case Studies: Pedagogical Approach, Guidelines, and Implementation" by Jaroslav Porubán, Milan Nosál', Matúš Sulir and Sergej Chodarev (contribution 7) proposes a novel method for teaching programming by combining task-driven learning with case study approaches. It is grounded in constructivism learning theory, emphasizing learning through solving real-world tasks rather than relying solely on theoretical instruction. The approach uses structured case studies (e.g., a programming project like Minesweeper) to guide students step-by-step in applying concepts and developing solutions. The authors provide practical guidelines for designing tasks, organizing courses, and supporting students during implementation. Findings

suggest that this method improves student engagement, understanding, and practical programming skills, as learners actively apply knowledge and learn from mistakes. Overall, the study concludes that task-driven case studies offer an effective, practice-oriented alternative to traditional programming instruction.

The article “Gamified Project-Based Learning in Vocational Education and Training Computer Science Courses” by Belkis Díaz-Lauzurica and David Moreno-Salinas (contribution 8) investigates how combining gamification with project-based learning (PBL) affects Computer Science Education. It uses a qualitative approach to analyze students’ learning experiences and engagement within vocational education settings. Findings show that gamified PBL increases student motivation, participation, and collaboration, making learning more interactive and engaging. Students also develop stronger problem-solving, algorithmic thinking, and practical programming skills through hands-on, project-oriented activities. However, the study highlights challenges such as the need for careful instructional design and teacher support to effectively integrate gamification elements. Overall, the approach is considered beneficial for enhancing learning outcomes and aligning Computer Science Education with real-world, skill-based demands.

The article “Measuring Undergraduates’ Motivation Levels When Learning to Program in Virtual Worlds” by Juan Gabriel López Solórzano, Christian Jonathan Ángel Rueda and Osslan Osiris Vergara Villegas (contribution 9) investigates the effectiveness of virtual-world environments as pedagogical tools for introductory programming education. It employs a quasi-experimental pre-test/post-test design, comparing undergraduate students’ motivation in traditional classroom settings with those learning in immersive virtual environments. Findings indicate that students exposed to virtual worlds demonstrate significantly higher levels of motivation than those taught through conventional methods, suggesting that immersive and interactive contexts can enhance engagement in programming education. The study also reports gender-related differences, with female students exhibiting greater confidence levels in the virtual learning condition. Methodologically, the research highlights the importance of integrating information and communication technologies (ICTs) into teaching strategies, while emphasizing the need for structured instructional design when adopting virtual environments. Overall, the article concludes that virtual worlds represent a promising complementary approach to traditional instruction, though their effectiveness depends on careful pedagogical implementation and alignment with course objectives.

5. Automatic Assignment Creation and Assessment

This thematic area consists of two papers, with the first one referring to the automatic construction of tests and assignments from large problem banks and the second one on automatic assessment tools that can be used both for exercise and project assessment.

The article “Searching Questions and Learning Problems in Large Problem Banks: Constructing Tests and Assignments on the Fly” by Oleg Sychev (contribution 10) addresses the challenge of generating effective assessments from large-scale repositories of automatically created or curated problems. It highlights that modern educational systems increasingly rely on extensive question banks, where manual inspection and selection by instructors become impractical. To address this issue, the study proposes an algorithm that constructs individualized tests dynamically based on teacher-defined constraints, such as required concepts and difficulty levels. The algorithm ensures that generated assignments meet pedagogical objectives by controlling the distribution of target concepts while adapting difficulty according to the available problem pool. Empirical evaluation on a large dataset demonstrates that the method reliably produces diverse and balanced exercises, reducing predictability and limiting opportunities for cheating through shared

solutions. Additionally, the results show that task difficulty is primarily influenced by the characteristics of the problem bank rather than parameter tuning alone. Overall, the article concludes that algorithmic generation of assessments can support scalable, individualized learning in contexts such as massive open online courses, provided that problem banks are well-structured and sufficiently rich in content.

The article “Bridging the Gap between Project-Oriented and Exercise-Oriented Automatic Assessment Tools” by Bruno Pereira Cipriano, Bernardo Baltazar, Nuno Fachada, Athanasios Vourvopoulos and Pedro Alves (contribution 11) examines limitations in existing automatic assessment tools (AATs) for programming education, particularly their difficulty in supporting both large-scale projects and small exercises within a unified framework. It introduces an IntelliJ IDEA plugin that extends the Drop Project AAT, enabling more efficient handling of short, exercise-based assignments while preserving the benefits of project-oriented assessment. Methodologically, the study combines system design with an empirical evaluation based on a student survey conducted during the 2022–23 academic year. Results indicate that the plugin improves usability, streamlines submission workflows, and integrates instructions and feedback directly within the development environment, thereby enhancing learning efficiency. Student responses suggest a generally positive perception, particularly regarding time savings and ease of use, although the small number of survey respondents limits the generalizability of findings. Such IDE-integrated tools can bridge the pedagogical divide between project-based and exercise-based learning approaches in Computer Science Education. The study concludes that bridging the gap between project-oriented and exercise-oriented AAT offers a promising direction for future educational technologies, provided they are iteratively refined through user feedback and broader evaluation.

6. Conclusions

Research in the field of Computer Programming Education has been at the forefront for several decades. Students’ difficulties and misconceptions with programming concepts have been thoroughly investigated but are still under investigation, with the aim of devising effective didactical interventions and tools, such as intelligent tutoring systems. Moreover, they are utilized in combination with other data and techniques for predicting performance at early stages, in order to support students and prevent them from dropping out. In addition, contemporary forms of Information and Communication Technologies such as digital twins and virtual worlds, as well as innovative teaching approaches or combinations of them, such as task-driven case studies and gamified project-based learning, have been proposed for supporting and motivating students in learning programming. Another field that has attracted researchers’ interest is the creation of tools that automate the process of devising assignments and tests from large problem banks and the process of assessing students’ assignments, tools that are considered important nowadays for supporting large audiences and massive open online courses. Finally, programming has been considered a valuable means of cultivating CT skills, which are considered so important for everyone that several educational systems around the world make efforts towards incorporating them into their Computer Science curriculum from early ages.

The articles published in this Special Issue have dealt with all the aforementioned challenges and have highlighted other challenges and future trends as well that require further research.

Funding: This research received no external funding.

Acknowledgments: We would like to thank all the authors and peer reviewers for their valuable contributions to the Special Issue on “Future Trends on Computer Programming Education”.

Conflicts of Interest: The author declares no conflicts of interest.

List of Contributions

1. Chen, L.-H.; Nguyen, H.T.T. Study Trends and Core Content Trends of Research on Enhancing Computational Thinking: An Incorporated Bibliometric and Content Analysis Based on the Scopus Database. *Computers* **2024**, *13*, 91. <https://doi.org/10.3390/computers13040091>.
2. Yiatas, D.; Jimoyiannis, A. Enacting Computer Science Curriculum Reform: The Case of Model and Experimental Lower Secondary Schools in Greece. *Computers* **2026**, *15*, 140. <https://doi.org/10.3390/computers15030140>.
3. Manzanares, M.J.; Pérez Marín, D.; Pizarro, C. A Preliminary Usability Study of a Novel Educational Training System to Teach ScratchJr. in School. *Computers* **2026**, *15*, 17. <https://doi.org/10.3390/computers15010017>.
4. Sychev, O.; Denisov, M. Explain Trace: Misconceptions of Control-Flow Statements. *Computers* **2023**, *12*, 192. <https://doi.org/10.3390/computers12100192>.
5. Pires, J.P.J.; Brito Correia, F.; Gomes, A.; Borges, A.R.; Bernardino, J. Predicting Student Performance in Introductory Programming Courses. *Computers* **2024**, *13*, 219. <https://doi.org/10.3390/computers13090219>.
6. Lu, M.; Hu, Z. Digital Twin-Enhanced Programming Education: An Empirical Study on Learning Engagement and Skill Acquisition. *Computers* **2025**, *14*, 322. <https://doi.org/10.3390/computers14080322>.
7. Porubán, J.; Nosál', M.; Sulír, M.; Chodarev, S. Teach Programming Using Task-Driven Case Studies: Pedagogical Approach, Guidelines, and Implementation. *Computers* **2024**, *13*, 221. <https://doi.org/10.3390/computers13090221>.
8. Díaz-Lauzurica, B.; Moreno-Salinas, D. Gamified Project-Based Learning in Vocational Education and Training Computer Science Courses. *Computers* **2026**, *15*, 82. <https://doi.org/10.3390/computers15020082>.
9. López Solórzano, J.G.; Ángel Rueda, C.J.; Vergara Villegas, O.O. Measuring Undergraduates' Motivation Levels When Learning to Program in Virtual Worlds. *Computers* **2024**, *13*, 188. <https://doi.org/10.3390/computers13080188>.
10. Sychev, O. Searching Questions and Learning Problems in Large Problem Banks: Constructing Tests and Assignments on the Fly. *Computers* **2024**, *13*, 144. <https://doi.org/10.3390/computers13060144>.
11. Cipriano, B.P.; Baltazar, B.; Fachada, N.; Vourvopoulos, A.; Alves, P. Bridging the Gap between Project-Oriented and Exercise-Oriented Automatic Assessment Tools. *Computers* **2024**, *13*, 162. <https://doi.org/10.3390/computers13070162>.

References

1. Robins, A.; Rountree, J.; Rountree, N. Learning and Teaching Programming: A Review and Discussion. *Comput. Sci. Educ.* **2003**, *13*, 137–172. [\[CrossRef\]](#)
2. Xinogalos, S. Designing and Deploying Programming Courses: Strategies, Tools, Difficulties and Pedagogy. *Educ. Inf. Technol.* **2016**, *21*, 559–588. [\[CrossRef\]](#)
3. Brusilovsky, P.; Calabrese, E.; Hvorecky, J.; Kouchnirenko, A.; Miller, P. Mini-Languages: A Way to Learn Programming Principles. *Educ. Inf. Technol.* **1997**, *2*, 65–83. [\[CrossRef\]](#)
4. Xinogalos, S.; Satratzemi, M.; Malliarakis, C. Microworlds, Games, Animations, Mobile Apps, Puzzle Editors and More: What Is Important for an Introductory Programming Environment? *Educ. Inf. Technol.* **2017**, *22*, 145–176. [\[CrossRef\]](#)
5. Xinogalos, S. Using Flowchart-Based Programming Environments for Simplifying Programming and Software Engineering Processes. In Proceedings of the 2013 IEEE Global Engineering Education Conference (EDUCON), Berlin, Germany, 13–15 March 2013; pp. 544–552. [\[CrossRef\]](#)
6. Wing, J.M. Computational Thinking. *Commun. ACM* **2006**, *49*, 33–35. [\[CrossRef\]](#)
7. Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Kafai, Y. Scratch: Programming for All. *Commun. ACM* **2009**, *52*, 60–67. [\[CrossRef\]](#)
8. Yaune, J.; Bartholomew, S.R.; Rich, P. A Systematic Review of “Hour of Code” Research. *Comput. Sci. Educ.* **2023**, *33*, 512–544. [\[CrossRef\]](#)
9. Du, J.; Wimmer, H.; Rada, R. Hour of Code: Can It Change Students' Attitudes toward Programming? *J. Inf. Technol. Educ. Innov. Pract.* **2016**, *15*, 53–70. [\[CrossRef\]](#) [\[PubMed\]](#)

10. Prensky, M. *Teaching Digital Natives: Partnering for Real Learning*; Corwin Press: Thousand Oaks, CA, USA, 2010.
11. Williams, L.A.; Kessler, R.R. All I Really Need to Know about Pair Programming I Learned in Kindergarten. *Commun. ACM* **2000**, *43*, 108–114. [[CrossRef](#)]
12. da Silva Estácio, B.J.; Prikładnicki, R. Distributed Pair Programming: A Systematic Literature Review. *Inf. Softw. Technol.* **2015**, *63*, 1–10. [[CrossRef](#)]
13. Satratzemi, M.; Xinogalos, S.; Tsompanoudi, D. Distributed Pair Programming in Higher Education: A Systematic Literature Review. *J. Educ. Comput. Res.* **2023**, *61*, 546–577. [[CrossRef](#)]
14. Raihan, N.; Siddiq, M.L.; Santos, J.C.; Zampieri, M. Large Language Models in Computer Science Education: A Systematic Literature Review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS '25)*, Pittsburgh, PA, USA, 26 February–1 March 2025; ACM: New York, NY, USA, 2025; pp. 938–944. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.