Teaching Computational Thinking Through a Cross-Curricular Approach Supported by Programming Patterns

Deller Ferreira[a], Cássio Martins[b], Samuel Costa[c] and Dirson Campos[d]
Institute of Informatics, Feredal University of Goiás, Campus Samambaia, Goiania, Brazil

Keywords: Computational Thinking, Cross-Curricular, Programming Patterns.

Abstract: Computational thinking means thinking or solving problems like computer scientists. It refers to the thought processes needed to understand problems and formulate solutions, making it a crucial skill for success in today's world. Therefore, it is essential that schools provide students with the necessary skills to think logically and solve problems. However, there is little knowledge among teachers about computational thinking, and some misconceptions about it suggest a demand for the term to be better explored in the context of initial teacher training. In this research, design-based research was used to develop teaching strategies and tasks for elementary students, involving programming patterns to develop computational thinking skills cross-curricularly. Six teachers positively evaluated a questionnaire analysing the strategies and tasks regarding clarity, compatibility, productivity, technological role, scope, and student focus. The set of cross-curricular teaching strategies involving programming patterns to develop thinking skills presented in this research constitutes an innovative and effective approach to teaching computational thinking in a contextualized, integrated, and systematic way.

1 INTRODUCTION

According to Hsu et al. (2018), research on computational thinking (CT) has increased over the last ten years. Teaching CT is a way to train students to be more than just consumers of technology. CT can be seen as a gathering of concepts and tools from computer science that are applicable in solving real-world problems. Integrating computational thinking into the curriculum can help students develop 21st-century skills such as creativity, critical thinking, and problem-solving.

CT (Computational Thinking) means thinking or solving problems like computer scientists. CT refers to the thought processes required to understand problems and formulate solutions. CT involves logic, evaluation, decomposition, automation, and generalization. According to Wing (2008), computational thinking is a type of analytical thinking. CT involves skills necessary to participate in the digital world and can be applied in various disciplines and contexts, including computer science, mathematics, sciences, and the humanities (Wing, 2006). Overall, computational thinking is a problem-solving process that emphasizes breaking down complex problems into smaller parts, recognizing patterns, developing algorithms, and using automation to solve problems across multiple domains.

CT is an interconnected set of skills and practices for solving complex problems, a way to learn topics in many disciplines, and a necessity for full participation in a computational world (Yadav et al., 2017). CT is seen as an important competency necessary for adapting to the future. However, educators, especially elementary school teachers and researchers, have not clearly identified how to teach it (HSU et al., 2018). Yadav et al. (2017) revealed that pre-service teachers without prior exposure to CT have a superficial understanding of computational thinking.

Despite several resources and tools being available to help educators integrate computational

[a] https://orcid.org/0000-0002-4314-494X
[b] https://orcid.org/0009-0005-8967-7134
[c] https://orcid.org/0009-0009-4356-6710
[d] https://orcid.org/0000-0002-0878-8336

Ferreira, D., Martins, C., Costa, S. and Campos, D.
Teaching Computational Thinking Through a Cross-Curricular Approach Supported by Programming Patterns.
DOI: 10.5220/0013137700003932

In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU 2025) - Volume 2*, pages 641-648
ISBN: 978-989-758-746-7; ISSN: 2184-5026

thinking into an interdisciplinary approach, there are challenges and opportunities to be uncovered in integrating it throughout elementary education, with promising practices and strategies to be discovered for moving computational thinking from concept to deep integration across different disciplines.

As seen, the importance of CT extends beyond computing, and it should be integrated into cross-curricular approaches. Due to the importance of CT in disciplines such as mathematics, social studies, language, and sciences (Wing, 2006), involving its underlying concepts, benefits, surrounding issues, forms of assessment of students' understanding of these concepts, and approaches for applying this concept in elementary education, as well as the great interest of researchers and educators on the subject, the motivation arose to develop a cross-curricular teaching approach.

A promising approach is the use of programming patterns as a cross-curricular strategy for teaching computational thinking. In this research, we consider a cross-curricular approach to be a teaching approach that spans different areas of knowledge present in the school curriculum. A programming pattern, simply put, is a way of solving a recurring problem, that is, a common solution for a particular problem (Proulx 2000).

In this context, the objective of the present research is to create teaching strategies for elementary students, involving programming patterns to develop computational thinking skills cross-curricularly.

2 LITERATURE REVIEW

Recent developments highlight the importance of developing interdisciplinary work skills, where students learn to meaningfully relate computational concepts across different disciplines (Celepkolu et al., 2020). The existing literature supports the inclusion of computational thinking (CT) in elementary school curricula across various subjects, starting from early childhood education. This approach requires students to learn how to use CT in ways that allow them to apply what they have learned to different domains (Chakarov et al., 2019).

From the student's perspective, despite its overall effectiveness, the transfer of skills between different subjects can be challenging, especially for younger students. An open challenge for computer science education researchers is to develop a deep understanding of the student experience in integrating CT across disciplines (Celepkolu et al., 2020). From the teacher's perspective, teaching CT in a cross-curricular manner can also be challenging. In Yadav et al.'s (2017) study, 134 pre-service teachers were asked about their views on computational thinking and its role in teaching CT in elementary classrooms. The goal of the research was to understand pre-service teachers' perceptions of CT in their specific areas and to assess how they would implement it in their future classrooms. The results indicated that elementary school teachers have only a superficial understanding of computational thinking.

Carvalho and Braga (2022) corroborate this result by noting that there is still little knowledge among teachers about CT, and some inadequate understandings suggest a need for the term to be better explored in the context of initial teacher education. Falcão and França (2021) also point out the lack of training in CT, tied to the low level of digital literacy among Brazilian teachers.

To meet these demands, the computer science education community has long been investigating best practices to prepare students for the essential skills needed in a computer-dependent world. Barr and Stephenson (2011) related computational thinking (CT) to various fields, identifying and exemplifying core CT concepts and strategies applied across different disciplines. For example, problem decomposition was mapped to science through species classification and to language instruction through outlining. The concept of abstraction was applied to language by writing a branching story, mapped to social studies by summarizing facts and drawing conclusions, and contextualized in physics by constructing a model of a physical entity.

In the research of Goldberg et al. (2012), elementary and middle school students were introduced to computational thinking and computer science concepts, including algorithms, graph theory, and simulations in interdisciplinary contexts, reflecting how computing technologies are used in research and industry. Computing was embedded into courses students were already taking, including art, biology, health education, mathematics, and social studies.

Souza and Menezes (2023) developed computational thinking strategies through a cross-curricular pedagogical framework for fifth-grade students, countering the skills of the National Common Core Curriculum in the areas of languages, natural sciences, and mathematics. For example, to explore everyday phenomena that demonstrate physical properties of materials, such as density, thermal and electrical conductivity, responses to magnetic forces, solubility, responses to mechanical forces, among others, they proposed the creation of

an algorithm to inquire about the material by asking: What is the product's consistency? Is it malleable or not? What is its resistance (is it degradable in a few days, weeks, months)? Where is it used?

Güven and Gulbahar (2020) provided assistance to social studies teachers regarding how CT can be integrated into elementary school classrooms. For example, to encourage students to think abstractly, they suggest assigning tasks such as creating a model of cultural changes, urbanization, and population growth. Regarding decomposition, teachers can ask students to identify the main reasons and consequences for differences in population distribution within and between countries. For algorithmic thinking, teachers can ask students how a certain bill became law. Students can study the steps necessary for a proposed bill to become law and then draw an algorithmic flowchart.

Ragonis and Shilo (2018) conducted a study where the results showed that students' understanding of argumentative texts improved after learning programming logic and the teacher applied an interdisciplinary learning facilitation technique through analogies between the structure components of an argument and commands in an algorithm.

A different way of thinking about CT is to move beyond its abstract definitions toward a more pragmatic conceptualization. Basawapatna et al. (2011) applied CT to science teaching through analogies between programming patterns and science simulations. Students and teachers who participated in a summer game development course were given a CT questionnaire. This questionnaire tested the participants' ability to recognize and understand patterns in a science context. They found that most participants were able to comprehend and recognize the patterns in various contexts.

Yadav et al. (2017) argue that there is a set of CT concepts that allow introducing CT concepts into other areas of knowledge; these concepts include data collection, data analysis, and data representation. According to the authors, a social studies teacher can use data, such as the most used words in presidential inaugural speeches over a period of time, and students can analyze the differences between speeches across time periods or between presidents of different parties. The ability to make sense of a dataset to solve a problem is a fundamental CT skill.

Lee et al. (2011) argued that CT shares elements with several other types of thinking, and one of these, according to the authors, is mathematical thinking. In their mapping, Barr and Stephenson (2011) associated CT with processes used in solving mathematical problems, such as performing long division or factorization, where each step can be guided by a logical and well-defined reasoning. The authors also associated certain skills that are practiced and developed through CT education and can be applied in mathematics, such as problem decomposition. For example, applying an order of operations in solving an expression.

CT has been offered as an interdisciplinary set of mental skills derived from the discipline of computer science. However, the approaches found in the literature lack an explicit correlation of other areas with computing and an interconnection between the different domains. The use of programming patterns allows for an integrated view of CT across different disciplines, thus facilitating the transfer of CT skills to distinct contexts.

Using patterns during the teaching and learning process allows students to accelerate the development of skills such as abstraction, problem decomposition, and identifying a recurring problem (Proulx, 2000), which are fundamental skills related to computational thinking. Besides these concepts, the research addresses cross-curricular integration, relating CT not only to computing but in an integrated manner.

A cross-curricular approach seeks to go beyond the existing space of each discipline in order to produce knowledge and learning, connecting learning to people's lives. Programming patterns can be instantiated in different disciplines, from the most common ones like mathematics to even physical education (Leal & Ferreira, 2016), thus allowing a single computational solution to be viewed from different perspectives. In other words, students can apply the same solution and think computationally across different disciplines in a uniform way.

There is little work on the teaching of programming patterns, and in the specialized literature, as far as we know, no work addresses the teaching of programming patterns combined with the teaching of computational thinking in a cross-curricular manner. Thus, this research represents a relevant and original contribution to CT education.

3 METHODOLOGY

This research used a qualitative research methodology based on the design-based research (DBR) method. The basic process of DBR involves developing solutions to problems. There are different ways to describe DBR found in the literature. In this work, the model chosen was that of Romero-Ariza (2014). The approach proposed by Romero-Ariza involves three main phases: a preliminary

investigation phase where the needs and the problem are clarified, a development and implementation phase, involving progressive improvement in iterative cycles of prototypes aimed at achieving the research objective, and a final evaluation phase to validate whether the result obtained is consistent with the defined objective.

3.1 Methodological Steps

For this research, DBR was applied to develop a set of strategies for teaching computational thinking in a cross-curricular manner in elementary education. The structure of the phases and steps of the research, aiming to meet the characteristics of the DBR approach according to the model proposed by Romero-Ariza, are described as follows:

Phase 1: Preliminary Investigation
Literature review on cross-curricular approaches to CT. Application of questionnaires to analyze teachers' familiarity with CT and cross-curricular approaches.

Phase 2: Development and Implementation
Initial development of a set of strategies using cross-curricular programming patterns. Definition of a context for cross-curricular application of the strategies. Initial development of tasks to implement the developed strategies. Application of a questionnaire to evaluate the tasks by teachers regarding efficiency and effectiveness. Reformulation of tasks in a participatory manner with teachers.

Phase 3: Final Evaluation
Application of a questionnaire to analyze the strategies and tasks by teachers in terms of their clarity, compatibility, productivity, technological role, scope, and student focus (Kimmons et al., 2020).

4 RESULTS

4.1 Strategies for Using Programming Patterns in a Cross-Curricular Manner

The main contribution of the research is the development, with the participation of teachers, of a set of strategies and tasks for teaching computational thinking (CT). From these strategies, a set of activities is created using a systematized and integrated approach to teach CT in a cross-curricular way. These activities consist of tasks developed for different subjects beyond computer science. The patterns used in this work can be found on the Elementary Patterns Home Page (Wallingford, 2001). As an example of a programming pattern, we have sequential choice. Sequential choice addresses a situation where exactly one of several possible actions must be chosen, but the action does not depend on the value of a single expression. Instead, suppose each action depends on a separate testable condition. All the subjects involved are addressed in a cross-curricular, integrated, and systematized manner. This integration occurs by contextualizing a real-world theme and its problematization in the subjects, as well as by using a common approach across subjects through programming patterns. Problematization is systematically presented through the application of strategies that have a common denominator, which is the programming pattern. This systematized integration allows students to visualize a cross-curricular application of CT, making the learning process more meaningful. The strategies are presented below.

4.1.1 Understanding Programming Patterns

In the understanding strategy, the activity begins in the computer lab and later continues in other subjects. In this work, the Scratch programming language was used. Understanding tasks are divided into two subtypes: tasks involving the use of patterns and tasks linked to anti-patterns. Here, anti-patterns are considered as common erroneous solutions that have a correct part.

By applying patterns in other subjects, students expand their understanding of a concept through analogies with other constructs. This strategy is related to overcoming and visualizing concepts and ideas in a broader way. Seeing an idea in different contexts and also seeing ideas in a larger scenario is a way to overcome conceptual barriers. Considering ideas in new contexts is a way of perceiving other possible uses and meanings. This type of task is related to the flexibility of divergent thinking.

The use of anti-patterns takes advantage of the way bad ideas become beneficial deviations for good ideas. Students do not only reflect on positive impacts, relevant implications, or good characteristics but also reflect on why a failure occurred, on the impacts, characteristics, and negative implications. They do not just eliminate the wrong paths but reflect and take advantage of them. Students transform ideas and concepts into new interpretations, also thinking about mistakes. Furthermore, anti-patterns can reflect partially correct solutions that are

associated with more simplistic thinking, making them easier for students to understand. From the student's understanding of the anti-pattern, the teacher moves on to a second explanation of how to correct it.

4.1.2 Recognizing Programming Patterns

The recognition strategy can be adopted in any subject and involves tasks in which students must identify one or more previously addressed patterns within a presented solution. The goal is for students to exercise the ability to identify situations where a pattern can be applied to solve a problem more quickly or improve a solution. In this type of task, analogical reasoning is applied to problem-solving.

Analogical reasoning is one of the most important problem-solving heuristics. It is related to transferring solutions from previously known problems to new ones and the ability to abstract similarities and apply productive past experiences to new situations. When students examine problems similar to familiar structures, they gain more robust conceptual knowledge about the problems, building a stronger problem schema.

4.1.3 Adapting Programming Patterns

The goal of the adaptation strategy is for students, having been introduced to content in previous classes, to deepen or enhance their knowledge of that content. In these classes, students will be challenged to adapt some activity, creating something different and new from what they have already seen and discussed. The adaptations can be minor or major. The adaptation strategy can be used in any subject.

This strategy is related to the divergent thinking skills of elaboration and fluency. Elaboration and fluency are two fundamental components of the creative process. The teacher can encourage students to improve these skills by making explicit what is already there but hidden, as well as dealing with the elements of who, what, why, and how of solution ideas. Students uncover opportunities by searching for attributes and relationships between concepts and new ideas, and they try to organize and reorganize the information.

4.1.4 Combining Programming Patterns

The combination strategy can be applied in any subject. In the combination strategy, the goal is for students to apply more than one pattern within a single solution. The way students can organize these patterns can be done sequentially or with one pattern as part of another.

This strategy is also related to the fluency skill, just like the adaptation strategy. Additionally, it is related to the problem-solving processes of decomposition and the "divide and conquer" paradigm. Simply put, problem decomposition aims to separate or divide a complex problem into smaller problems, making each problem's solution easier. Hence, the idea of "divide and conquer" comes into play. It is a paradigm that breaks a complex problem into small subproblems, and after solving each subproblem, the solutions are combined to solve the initial problem.

4.2 Tasks Based on Strategies for Using Programming Patterns in a Cross-Curricular Manner

The chosen context for cross-curricular application of the strategies was COVID-19. The theme of COVID-19 was used to outline various problems for students to solve in different subjects. 24 tasks were developed as practices for applying patterns in a cross-curricular way, four for each involved subject, which were computer science, science, physical education, social studies, languages, and mathematics. Below are six sample tasks.

The moving average is a tool that helps to understand how COVID is behaving. It is calculated by summing the number of cases from the last 7 days, and after summing, it is necessary to divide this amount by 7. As an example, see Table 1.

Table 1: Evolution of COVID-19 Case Numbers.

Day of the Week	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Number of cases	10	20	25	30	38	45	56

In this presented scenario, the moving average for Sunday is the sum of the number of cases from the last 7 days:

$S = 10 + 20 + 25 + 30 + 38 + 45 + 56$
$S = 224$

Division of the sum by 7:
$M=224/7$
$M = 32$

The teacher should clarify what the moving average is by discussing questions such as:

- Why should we sum all the values?
- Why should we divide by 7?

The teacher should discuss questions that make sense of the formula used to calculate the moving average.

After introducing and discussing the concept and formula of the moving average, ask the students to write a sequence of instructions to calculate the moving average of the number of COVID cases over the last 7 days. The accumulator pattern should be adapted and combined with the sequential pattern when writing the instructions.

After constructing the sequence, students should execute it and discuss its operation.

4.3 Interaction with Teachers

Six elementary school teachers from the subjects of informatics, mathematics, sciences, social studies, physical education, and languages participated in the research.

4.3.1 Precedents and Context Analysis

41.7% of teachers know or have used programming or computational thinking (CT) terms. 41.7% know or have used these terms to a limited extent. 16.6% do not know or have not used programming or CT terms.

41.7% were familiar with CT or programming terms, 41.7% had little knowledge of them, and 16.6% were unfamiliar with the terms.

Even though 58.3% had at least some knowledge of the terms, only 41.7% had experience with programming and CT in class. Of these, only those in the informatics discipline had more solid and significant experiences with CT and programming in teaching, using educational software to teach basic computer principles. Still, all teachers, even briefly introduced to CT, showed interest in working with the concept in their subjects.

For CT integration to occur effectively, the teacher must be motivated and engaged in using computational thinking in their elementary school subject. The results presented a predisposition to seek new pedagogical strategies using CT that can enrich student learning.

In the set of practices proposed for this research, one concept used to make CT understanding and learning more meaningful is transversality. 60.8% of teachers had experiences with transversal teaching approaches, while 33.2% had not. 60.8% of teachers were interested in applying transversal approaches, and 16.6% were not.

The teachers reported what they knew and their experiences with transversal teaching approaches. Of the total participants, 60.8% had experiences with transversal teaching approaches, but not all had positive experiences. Some reported that in their experiences, they participated in groups with students from different grade levels, and the disparity between knowledge levels limited teamwork.

Regarding interest in working with transversal approaches, 83.4% declared interest. 16.6% did not state whether they were interested or not, as they were unfamiliar and had no experience with transversal approaches. The results showed that teachers are motivated to promote more integrated learning that connects different areas of knowledge and allows students to have a broader and more complex understanding of the content.

4.3.2 Formative Evaluation of Activities

A formative evaluation of the set of tasks for teaching CT was conducted. Based on the data collected from the precedents and context analysis, researchers and teachers proposed a set of requirements to guide the evaluation of the activities. This set of requirements is presented in Table 2.

Table 2: Transversality Precedents.

Regarding the set of tasks	Regarding individual tasks
- Does it teach and encourage the practice of CT?	- Does it address CT?
- Is it transversal?	- Does it address patterns and/or anti-patterns?
- Were the types of tasks explored?	- Does it explain the concepts it will cover?
- Does it track the continuous and cumulative evolution of student performance?	- Does it engage with the cognitive capacity of elementary school students?

After the teachers studied and analyzed the set of activities, they identified which requirements were met and which still needed to be achieved. After this evaluation, the researchers improved the set to meet the unmet requirements, and then a new evaluation was conducted. This cycle was repeated twice until all the requirements were fulfilled for the set of activities.

4.3.3 Final Evaluation of the Strategies and Activities

Once the refinement cycles of the proposed set of practices were completed, all participating teachers were invited to answer a questionnaire for the final evaluation of the activities. 83.4% of the teachers considered the strategies for applying computational

thinking (CT) in their discipline sufficiently simple. 100% considered the strategies for applying CT in their discipline clear. 83.4% did not consider that the strategies for applying CT in their discipline had hidden complexities. 100% considered that the strategies for applying CT in their discipline complement valuable existing educational practices. 100% considered that the strategies for applying CT in their discipline support valuable existing educational practices. 100% considered that the strategies for applying CT in their discipline foster productive thoughts as teachers struggle with technology integration issues. 83.4% considered that the strategies for transversal application of CT reduce technology integration issues in their discipline. 83.4% did not consider that the strategies for transversal application of CT in their discipline were an end in themselves. 83.4% considered the strategies for transversal application of CT sufficiently parsimonious to ignore irrelevant aspects of technology integration. 83.4% considered the strategies for transversal application of CT in their discipline sufficiently comprehensive to guide their practice. 100% considered that the strategies for transversal application of CT in their discipline emphasize active student participation. 100% considered that the strategies for transversal application of CT could improve student outcomes in their discipline. 100% could visualize the patterns in other problems and contextualized situations in their discipline beyond those presented.

Based on the data presented, it can be concluded that the evaluation was positive in terms of clarity, compatibility, productivity, technological role, scope, student focus, and replicability.

5 CONCLUSIONS

Due to the importance of CT in subjects such as mathematics, social studies, language, and physical education in primary education, as well as the great interest of researchers and educators in the topic, it is essential to integrate transversal approaches to teach CT.

In this research, teaching strategies were developed for primary school students, involving programming patterns to develop computational thinking skills in a transversal way. This contribution is a relevant and original one in teaching CT, as it involves strategies and tasks that allow its application in an integrated and systematic way across different subjects.

The methodology used for developing the research was Design-Based Research (DBR). DBR offers a set of methods and methodological steps when building educational artifacts.

Questionnaires were used to evaluate the strategies by six teachers, one from each discipline covered: computer science, mathematics, languages, social studies, science, and physical education.

The results of the questionnaires showed that most participating teachers considered the strategies for applying CT in their discipline sufficiently simple, clear, and complementary to existing educational practices. Additionally, most believe that the strategies for transversal application of CT foster productive thoughts and can improve student outcomes in their discipline. Most also consider the strategies sufficiently parsimonious and comprehensive to guide their practice, emphasizing active student participation. However, a minority believes that the strategies have hidden complexities and are not an end in themselves. Furthermore, some indicated that the strategies for transversal application of CT may not reduce all technology integration issues in their discipline. Finally, all teachers can visualize the patterns in other problems and contextualized situations in their discipline beyond those presented.

The set of transversal teaching strategies involving programming patterns to develop CT skills presented in this research represents an innovative and effective approach to teaching CT skills in a practical and contextualized manner.

REFERENCES

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 245-250). https://doi.org/10.1145/1953163.1953280

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20-23. https://doi.org/10.1002/LLQT.20078

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Carvalho, F., & Braga, M. (2022). Pensamento computacional na educação brasileira: Um olhar segundo artigos do Congresso Brasileiro de Informática na Educação. *Revista Brasileira de Informática na*

Educação, 30, 237-261. https://doi.org/10.5753/rbie.2022.2649

Chakarov, A. G., Recker, M., Jacobs, J., Van Horne, K., & Sumner, T. (2019). Designing a middle school science curriculum that integrates computational thinking and sensor technology. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education - SIGCSE '19* (pp. 818-824). https://doi.org/10.1145/3287324.3287476

Celepkolu, M., Fussell, D. A., Galdo, A. C., Boyer, K. E., Wiebe, E. N., Mott, B. W., & Lester, J. C. (2020). Exploring middle school students' reflections on the infusion of CS into science classrooms. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)* (pp. 671-677). https://doi.org/10.1145/3328778.3366871

Pontual Falcão, T., & França, R. S. de. (2021). Computational thinking goes to school: Implications for teacher education in Brazil. *Revista Brasileira de Informática na Educação, 29*, 1158-1177. https://doi.org/10.5753/rbie.2021.2121

Goldberg, D. S., Grunwald, D., Lewis, C., Feld, J. A., & Hug, S. (2012). Engaging computer science in traditional education: The ECSITE project. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '12)* (pp. 351-356). https://doi.org/10.1145/2325296.2325377

Güven, I., & Gulbahar, Y. (2020). Integrating computational thinking into social studies. *The Social Studies, 111*(5), 234-248. https://doi.org/10.1080/00377996.2020.1749017

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.007

Kimmons, R., Graham, C. R., & West, R. E. (2020). The Picrat model for technology integration in teacher preparation. *Contemporary Issues in Technology and Teacher Education, 20*(1), 176-198. https://doi.org/10.30957/icitte20(1)rgw03

Leal, A. V., & Ferreira, D. J. (2016). Learning programming patterns using games. *International Journal of Information and Communication Technology Education (IJICTE), 12*(2), 23-34. https://doi.org/10.4018/IJICTE.2016040103

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Proulx, V. K. (2000). Programming patterns and design patterns in the introductory computer science course. *ACM SIGCSE Bulletin, 32*(1), 80-84. https://doi.org/10.1145/331383.331432

Ragonis, N., & Shilo, G. (2018). Analogies between logic programming and linguistics for developing students' understanding of argumentation texts. *Journal of Information Technology Education: Research, 17*(1), 549-575. https://doi.org/10.28945/3971

Romero-Ariza, M. (2014). Uniendo investigación, política y práctica educativas: DBR, desafíos y oportunidades. *Magis, Revista Internacional de Investigación en Educación, 7*(14), 159-176.

Souza, P. M., & Meneses, C. S. (2023). Uma arquitetura pedagógica para o desenvolvimento do pensamento computacional em contexto interdisciplinar. *RENOTE, 20*(2), 290-300. https://doi.org/10.22456/1679-1916.129185

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55-62. https://doi.org/10.1145/2994581

Wallingford, E. (2001). The elementary patterns home page. https://www.cs.uni.edu/~wallingf/patterns/elementary/

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3733.