

The Impact of Visual Programming Platforms (Scratch) on Developing Computational Thinking and Learning Skills in Schools

Problem Statement

Although computational thinking has become an essential skill for today's students, the rapid expansion of teaching visual programming environments—such as Scratch—in elementary and middle schools has not been matched by sufficient empirical research. Educational systems often operate on the assumption that these software tools automatically foster sub-skills like abstraction, decomposition, and algorithmic thinking; however, current evidence in the field remains contradictory and fragmented. For instance, Djambong and Freiman (2016) observed negligible cognitive gains among sixth-grade students using Scratch, noting a lack of any significant development in algorithmic thinking skills.

Similarly, Koray and Bilgin (2023) found that programming activities enhance collaboration and creativity but have no significant effect on problem solving or critical thinking. This discrepancy indicates a real gap in our knowledge: we do not know exactly which sub-skills Scratch develops and we do not know the most appropriate instructional environment for Scratch implementation.

This ambiguity presents a dilemma for curriculum developers in that the difference between exposing students to programming and developing genuine computational competence is sometimes blurred. To tackle this problem, recent studies have investigated how to relate Scratch to structured pedagogical frameworks. For example, Fagerlund et al. (2021) synthesized the field into 12 core educational principles to standardize teaching and assessment. Other authors have applied established instructional design models to ensure a coherent educational structure, such as the ADDIE model (Yusrina et al., 2026) or the 5E learning cycle (Koray & Bilgin, 2023).

In terms of assessment, some studies have attempted to assess differences through standardized tests such as Computational Thinking Test for Beginners (Piedade and Dorotea, 2023; Martínez-Murciano and Pérez-Jorge, 2024). However, extensive reviews show that despite such attempts, primary schools still lack flexible assessment tools that would be able to measure qualitative indicators – such as the ability to process information and creativity, since assessment is often confined to a superficial examination of the final project, without tracking the student's cognitive processes during the work process itself.

Moreover, there are obvious methodological problems with current research. Most research is conducted on small, non-random samples and in single schools over short periods of time (ranging from as little as four to eight weeks) which does not allow generalization of findings. In addition,

reliance on post-testing alone obscures basic individual differences among students present before the experiment. There is, therefore, an urgent need to synthesize this fragmented data, to explore the limitations of existing research, and to identify truly effective teaching practices. This study thus provides a detailed review and critique of the literature on teaching computational thinking through Scratch in schools.

Research Objectives

The goal of this study is to understand the current contradictions surrounding Scratch-based instruction, and to identify the computational skills that can be reliably developed and the most suitable educational environments for them. The study, to achieve this, is built on the following five main objectives:

1. To assess the effectiveness of Scratch programs in developing computational thinking skills (such as abstraction and decomposition) through reviewing the findings of previous quasi-experimental studies and systematic reviews.
2. Review of existing assessment tools used in primary schools (e.g., Computational Thinking Test for Beginners, Computational Thinking Scale) to evaluate their validity and to pinpoint their limitations, particularly in evaluating non-linear skills like creativity and collaboration.
3. Comparing the educational effectiveness of embedding Scratch in structured pedagogical frameworks (e.g., 5E and ADDIE models) versus teaching it as an independent programming subject.
4. Looking at the impact of age and other technological tools (e.g. robots) on the findings, considering the methodological limitations to the generalizability of the current research findings such as the duration and scale of the trials.

Literature Review

Summary of Research Papers

Paper 1: Task-Based Assessment of Students' Computational Thinking Skills Developed Through Visual Programming or Tangible Coding Environments

The first study (Djambong & Freiman, 2016) followed the development of computational thinking of 24 students in Canada. Ten 6th-grade students learned to program with Scratch for five weeks, and fourteen 9th-grade students worked with LEGO EV3 robots. The results of the pre- and post-tests showed only a very slight improvement in the skills (29.6% to 33.0% for the 6th graders and 44.9% to 48.2% for the 9th graders) and no development of algorithmic thinking. The study is limited by a small non-randomly selected sample, and the simultaneous mix of two variables (change of the technological tool and change of the age group).

Paper 2: Computational Thinking in Programming with Scratch in Primary Schools: A Systematic Review

The second study (Fagerlund et al., 2021) was a systematic review of 30 studies (from an initial pool of 425) that identified methods for evaluating computational thinking using Scratch from kindergarten to ninth grade. The researchers proposed a framework of 12 educational principles. they found that there was large variation in the methods of assessment (from analysis of a project to self-report measures) and that creativity and competence are rarely measured accurately, suggesting that there is no standardized, comprehensive assessment tool for the primary education stage.

Paper 3: Effects of Scratch-Based Activities on 4th-Grade Students' Computational Thinking Skills

Piedade and Dorotea conducted a quasi-experimental study in 2023 with 189 fourth-grade students in Portugal, using the standardized test for beginners. The group performed better on the final exam than any other group, suggesting that they had an advantage in understanding loops and if-then statements.

Paper 4: Computational Thinking with Scratch or App Inventor in Primary Education

Martnez-Murciano & Pérez-Jorge, 2024, conducted a systematic review using the PRISMA framework of ten studies that compare the use of Scratch and App Inventor in primary education. The evaluation revealed that the two tools promote a range of cross-curricular abilities; however, Scratch is more concerned with mathematical reasoning, while App Inventor is better at encouraging collaboration and creativity. The comparison is still incomplete as only one research study directly and extensively analyzed App Inventor.

Paper 5: CT Math - Enhancing Elementary Mathematics and Computational Thinking Skills through a Structured Scratch-Based Model

"CTMath" was introduced as a model for integrating computational thinking into primary school mathematics teaching, following the ADDIE instructional model, in the fifth study (Yusrina et al, 2026). A total of 13 experts validated the model and 43 students participated, with 22 allocated to the experimental group and 21 to enrolled group. The experimental group demonstrated significant improvement in their ability to identify and solve problems, as well as a strong grasp of plane geometry, despite the short duration of the experiment and its limited nature.

Paper 6: The Effect of Block Coding (Scratch) Activities Integrated into the 5E Learning Model in Science Teaching on Students' Computational Thinking Skills and Programming Self-Efficacy

This study, conducted over ten weeks, involved 22 sixth-grade students from Turkey who used Scratch activities as part of the 5E learning cycle in a science class. While computational thinking skills improved, only cooperation and creative skills did, with no substantial improvement in problem-solving, algorithmic thinking, or critical thinking. Furthermore, the lack of a control group and the small sample size significantly limit the generalizability of these findings.

Critical Analysis

Collectively, these six studies show that visual programming languages (primarily Scratch) are a powerful and promising tool for teaching computational thinking to children and young people. Still, a conclusive scientific conclusion must be reached on account of the methodological errors identified in this work.

Studies such as Piedade & Dorotea (2023) and Yusrina (2026) demonstrate a quasi-experimental methodology with control groups and standardized tests, showing clear statistical differences, which is consistent in its strengths. Moreover, the comprehensive reviews by Fagerlund (2021) and Martínez (2024) successfully provide an explicit map of the current literature.

. Studies by Koray & Bilgin and Yusrina suggest that programming has transformed from a mere technical activity to facilitated instruction through the use of explicit instructional models such as 5E and ADDIE.

On the other hand, there are some methodological deficiencies in this literature; the majority of samples are very small (such as the 22-student sample in the Djambong and Korai studies), and genuine random sampling is almost non-existent. Some studies do not conduct pre-tests, which undermines the precision of post-tested outcomes. In addition, conducting tests in a single school for 1-2 months raises questions about the students' ability to retain these skills over time.

Present-day assessment systems are incapable of evaluating the cognitive processes and creative thinking involved in programming for children through conventional multiple-choice tests or solely relying on the final outcome. In the future, research should focus on conducting long-term studies and utilizing dynamic assessment tools to capture the student's thought process at every step.

Connections

1. The shared tool 'Scratch' platform as primary teaching tool for all research.
2. Collectively, the papers aimed to gauge computational thinking by its most famous skills, including abstraction and decomposition, algorithms, debugging, and pattern recognition.
3. Research focused on a nearly equal grade range (4th, 5TH) and educational trends (7-9), suggesting that these years are the most conducive to learning computational algorithms.
4. Adhering to topics: Various studies have begun to connect programming with science and mathematics, rather than teaching it alone.

Conflicts Between the Findings

1. Various advanced skills were identified, for example, the work of Piedade & Dorotea and Yusrina demonstrated significant improvements in understanding conditional statements and iteration while the research conducted by Koray and Bilgin showed no improvement in algorithmic thinking or problem-solving.
2. Djambong & Freiman suggest that Scratch is not as effective as physical tools, but it can be used to enhance the performance of older students with tangible tools like Lego kits and robots. However, some studies indicate that only Scratch can produce satisfactory results.
3. Some argue that it's better to use Scratch in core subject lessons like math and science, while others see it as a standalone programming lesson. The issue is still unresolved as both methods have been successful.
4. A clear discrepancy exists between empirical studies that exclusively use standardized multiple-choice tests and systematic reviews that denounce these tools as insufficient in measuring computational thinking.

References

1. Djambong, T., & Freiman, V. (2016). Task-based assessment of students' computational thinking skills developed through visual programming or tangible coding environments. In 13th International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2016) (pp. 41–51). <https://competi.ca/wp-content/uploads/2014/04/publication-djambong-et-freiman-celda2016>.
2. Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12–28. <https://doi.org/10.1002/cae.22255>
3. Koray, A., & Bilgin, E. (2023). The effect of block coding (Scratch) activities integrated into the 5E learning model in science teaching on students' computational thinking skills and programming self-efficacy. *Science Insights Education Frontiers*, 18(1), 2825–2845. <https://doi.org/10.15354/sief.23.or410>
4. Martínez-Murciano, M. C., & Pérez-Jorge, D. (2024). Computational thinking with Scratch or App Inventor in primary education. *Campus Virtuales*, 13(2), 107–122. <https://doi.org/10.54988/cv.2024.2.1364>
5. Piedade, J., & Dorotea, N. (2023). Effects of Scratch-based activities on 4th-grade students' computational thinking skills. *Informatics in Education*, 22(3), 499–523. <https://doi.org/10.15388/infedu.2023.19>
6. Yusrina, H., Sutama, Sutopo, A., & Muhibbin, A. (2026). CTMath: Enhancing elementary mathematics and computational thinking skills through a structured Scratch-based model. *Frontiers in Education*, 11, 1767445. <https://doi.org/10.3389/feduc.2026.1767445>