Journal of Pedagogy and Education Science
Vol. 5, No. 03, pp. 879-894
journal.iistr.org/index.php/JPES
DOI: 10.56741/IISTR.jpes.002191



Integrating Scratch-Based Visual Programming to Enhance Analytical Thinking in ICT Education: A Systematic Review

[1]Risnawati, [1]Aris Budianto*, [2]Dedi Kuswadi
Corresponding Author: * arisbudianto@staff.uns.ac.id
[1] Universitas Sebelas Maret, Surakarta, Indonesia
[2] Universitas Negeri Malang, Malang, Indonesia

ARTICLE INFO

Article history
Received 23 March 2026
Revised 13 July 2026
Accepted 20 July 2026

ABSTRACT

This systematic literature review examined the potential of Scratch as a visual programming environment for strengthening students' analytical thinking in information and communication technology (ICT) education, particularly logical reasoning, problem decomposition, and evaluative judgment. Although previous research has widely investigated technical competence, engagement, creativity, and computational thinking, analytical thinking has rarely been treated as the primary learning outcome. Following the PRISMA protocol, searches of Scopus and ScienceDirect identified 1,381 records. After applying predefined inclusion and exclusion criteria, 16 peer-reviewed studies published in English between 2020 and 2025 were included in the qualitative synthesis. The findings indicate that Scratch's visual, block-based interface helps learners understand programming concepts, construct logical sequences, decompose complex problems, test alternatives, and evaluate solutions through iterative design. Scratch-based activities also encourage active participation, experimentation, creativity, feedback, and revision, thereby supporting higher-order cognitive development. The review contributes a structured account of how Scratch can be positioned not merely as an introductory coding tool but as a medium for cultivating analytical thinking. However, restricting the search to two databases, English-language publications, and the 2020–2025 period may have excluded relevant international evidence. The findings support systematically integrating Scratch into primary and secondary ICT curricula, especially beginner-level programming, through tasks that explicitly combine problem

decomposition, logical reasoning, solution comparison, debugging, reflection, and iterative improvement. Such integration should be accompanied by teacher guidance and assessment strategies that capture reasoning processes rather than focusing exclusively on functional program completion or technical accuracy.

Keywords
Analytical Thinking Skills
Computational Thinking
Scratch Programming
Programming Learning
Visual Programming

This is an open-access article under the CC–BY-SA license.

Introduction

According to the Organisation for Economic Co-operation and Development (OECD), analytical thinking, problem-solving, and complex reasoning are among the most critical competencies required in the modern workforce [1]. Thinking is defined as a higher-order cognitive skill that encompasses the ability to decompose complex problems into manageable parts, identify logical relationships between elements, and evaluate information systematically to reach reasoned conclusions [2]. This capacity enables learners to move beyond surface-level understanding, process information critically, and make decisions logically [3]. Unlike general critical thinking — which focuses broadly on evaluating arguments and assumptions — analytical thinking specifically involves structured decomposition, classification of information, comparison of solutions, and evaluative judgment within a given domain [2],[3]. As such, analytical thinking constitutes a foundational competency for navigating complex challenges in the 21st century, and its cultivation has become a central concern in modern educational curricula [1], [4].

Numerous studies indicate that students' analytical thinking abilities remain relatively low. For instance, Ardianti et al. [5] found that 49% of students fell into the very low category and 42% into the low category of analytical thinking skills. Similarly, Suyatman et al. [6] report that students' analytical thinking skills were still low based on data with three indicators of analytical thinking skills: differentiating (16.6), organizing (46.6), and attributing (7.2), based on a 0–100 rating scale. Other studies also reveal that evidence from various studies indicates that the analytical thinking skills of elementary school students remain at a relatively low level [7].

Beyond Scratch-based learning, other intervention studies have similarly demonstrated that multimedia-based media can strengthen students' analytical and problem-solving abilities. For instance, animated cartoons [8], Powtoon-based media [8], and

animation/manga materials [9] have each been shown to improve students' analytical thinking and problem-solving capabilities, while similar evidence underscores the broader relevance of analytical thinking in mathematics [11]and science education [12].

Analytical thinking skills need to be improved to solve problems in programming. Computational thinking and analytical thinking are core skills in programming because they help analyze and solve complex problems[12], particularly in programming and STEM contexts. Programming is one of the most difficult subjects for students, and many students struggle to grasp programming concepts [13]. Many students struggle to grasp basic programming concepts and develop strategies to solve problems. Students often struggle to grasp abstract programming concepts and the problem-solving process [14]. One of the media frequently used in the learning process to teach programming is Scratch. Scratch is a block-based visual programming language designed to facilitate students' creative and exploratory understanding of basic programming. Visual learning tools like Scratch [15] not only facilitate the understanding of important concepts such as variables, sequences, and conditional statements but also increase student interest and engagement.

Previous systematic literature reviews have extensively explored the role of Scratch in enhancing computational thinking and programming skills [16], problem-solving skills, and student engagement at various levels of education. For instance, Scratch has been widely recognized as an effective tool for developing computational thinking and programming competencies, particularly in K–12 education [17]. However, despite the growing body of literature, most studies primarily focus on computational thinking as the main outcome. In contrast, the explicit role of Scratch in fostering analytical thinking as a distinct cognitive domain remains underexplored. This gap is particularly significant, given that analytical thinking encompasses deeper cognitive processes such as evaluation, reasoning, and structured problem decomposition, which extend beyond procedural or technical programming skills [18]. In addition, previous reviews indicate a lack of clarity regarding how Scratch-based learning can be systematically integrated into educational curricula to support higher-order cognitive development [16]. Overall, research indicates that multimedia-based interventions and active learning approaches can improve analytical thinking skills; however, relatively few studies have specifically examined the contribution of visual programming platforms such as Scratch to this domain.

Therefore, to address these gaps, this study contributes by systematically synthesizing existing research on Scratch-based visual programming and explicitly positioning analytical thinking in its dimensions of logical reasoning, problem decomposition, and evaluative judgment as a primary outcome of programming learning. Furthermore, this study identifies key pedagogical and technological factors that support the development of analytical thinking

and proposes a structured approach for integrating Scratch into ICT curricula to enhance higher-order cognitive skills. Formulates. Accordingly, this study formulates the following research questions:

- RQ1: How does Scratch-based visual programming influence students' analytical thinking skills in ICT programming education?
- RQ2: What pedagogical and technological factors contribute to the development of students' analytical thinking skills when using Scratch in programming learning?
- RQ3: How can Scratch-based learning be integrated into ICT curricula to enhance analytical thinking as a higher-order cognitive skill?

Material and Method

This study adopted a systematic literature review (SLR) approach following the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) protocol to ensure transparency and methodological rigor [19]. PRISMA provides standardized guidelines that help researchers systematically identify, select, appraise, and synthesize relevant studies, thereby enabling readers to assess the trustworthiness and reproducibility of the review findings [20]. PRISMA was developed to address limitations in previous reporting practices and to incorporate advances in systematic review methodologies, ensuring that reviews present a clear and accurate account of why the study was conducted, how the evidence was collected, and what results were obtained [21]. PRISMA minimizes bias in evidence identification [19], but also improves the credibility of the findings for researchers, educators, and policymakers.

After formulating the research question, the next step is rigorous and systematic keyword selection. This determination is carried out to develop an effective search strategy using the selected databases. The accuracy and relevance of the selected keywords are critical for guaranteeing the comprehensive and unbiased identification of relevant studies. This review utilized the Scopus and ScienceDirect indexing databases, and the search was conducted between January and March 2025, covering publications from 2020 to 2025. The following Boolean search strings were applied across the title, abstract, and keyword fields: "Scratch" AND "analytical thinking"; "visual programming" AND "analytical thinking" AND "students"; "Scratch programming" AND "higher-order thinking"; "block-based programming" AND "problem decomposition" AND "education"; "student programming skills" AND "cognitive development". These queries were applied independently to each database, and the results were subsequently merged. Duplicate removal was performed manually by cross-checking article titles, authors, and publication years across both databases before screening commenced.

The inclusion and exclusion criteria were defined to ensure the relevance and quality

of the selected studies. Inclusion criteria were: peer-reviewed articles published between 2020–2025; written in English; focused on Scratch-based or visual programming in educational contexts; explicitly addressing cognitive outcomes such as analytical or computational thinking; available in full text. Exclusion criteria were operationalized as follows: (1) articles that do not specifically discuss Scratch or block-based visual programming as the primary learning tool; (2) non-empirical studies, including purely theoretical papers, editorials, book chapters, and opinion pieces; (3) articles not available in full text through the databases or institutional access; (4) studies not addressing learning outcomes related to cognition, thinking skills, or programming achievement; (5) non-English language publications; (6) studies conducted in contexts unrelated to programming or ICT education. The two databases yielded a total of 1,381 publications: 928 items from ScienceDirect and 453 articles from Scopus. Following duplicate removal, 1,294 articles were excluded from the data based on title and abstract selection, as they did not align with the SLR objectives. Ninety-six articles were submitted to the full paper review process. From the full paper review, 80 articles were excluded due to their unrelatedness to the research topic. Ultimately, 16 papers were included. The PRISMA screening process is shown in Fig. 1.

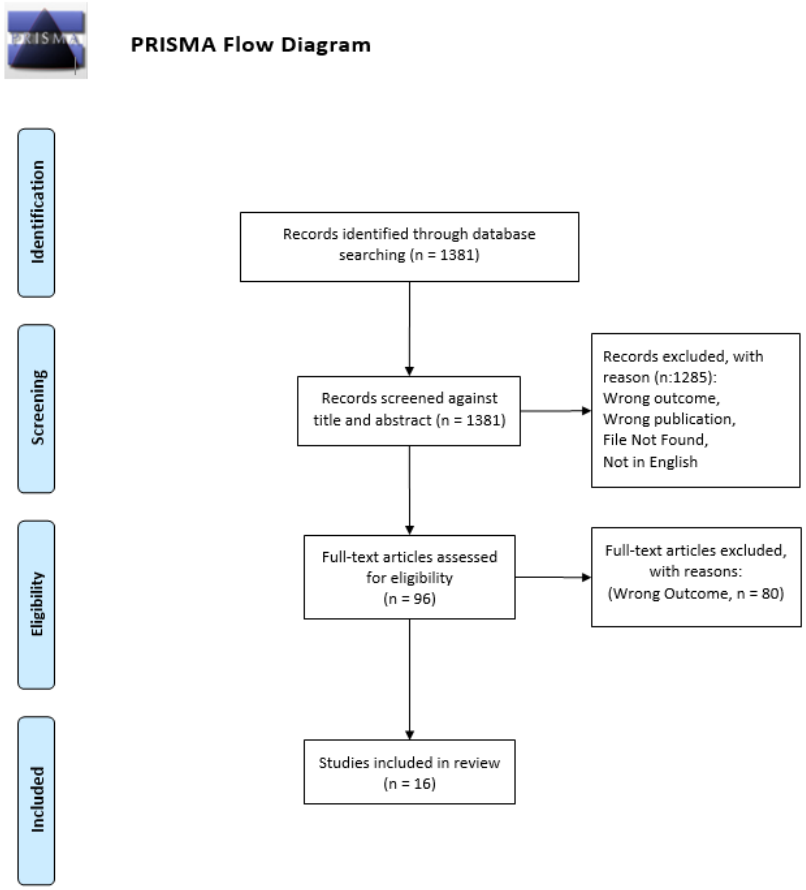


Fig. 1. PRISMA Flow Diagram

Publication trends show a consistent increase from 2020 to 2025. Our analysis shows that most Scratch-related research is more strongly correlated with improvements in computational thinking. Fewer articles discuss analytical thinking. In this analysis, we found 16 articles related to both analytical and computational thinking. This is because, in certain contexts, computational thinking is a subset of analytical thinking. The distribution of publications by year is shown in Fig. 2.

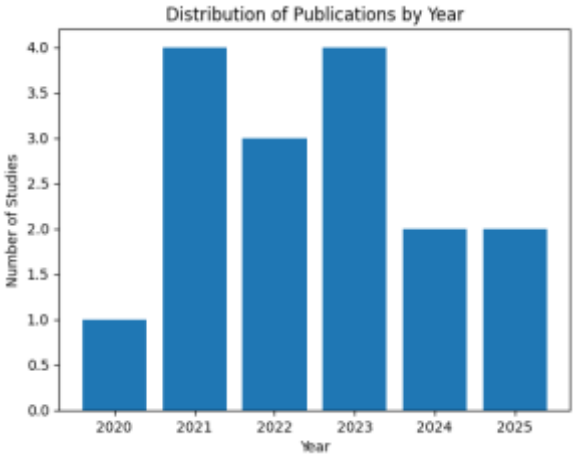


Fig. 2. Distribution of Publications by Year

Of the 16 articles, the distribution of publications was dominated by 15 journal articles (93.75%) and one conference proceeding (6.25%). In terms of indexing status, 13 articles were published in Scopus-indexed journals, including internationally recognized outlets published by Elsevier and Springer such as *Computers in Human Behavior*, *Education and Information Technologies*, *Thinking Skills and Creativity*, *Educational Technology Research and Development*, *Journal of Computers in Education*, *International Journal of Child-Computer Interaction*, *International Journal of Educational Methodology*, and *Journal of Educational Technology Development and Exchange*. One article was published in an open-access Scopus-indexed journal (*Education Sciences*, MDPI). One article appeared in the ACM International Conference Proceedings Series, which is indexed in Scopus as a conference venue. One article was published in the Asian Journal for Mathematics Education (SAGE), an open-access peer-reviewed journal indexed in DOAJ and ERIC. No articles were sourced from SINTA-indexed or nationally accredited journals, as this review focused exclusively on international databases (Scopus and ScienceDirect) to ensure broad international coverage.

This distribution indicates that the literature base reviewed in this study is predominantly sourced from internationally recognized, peer-reviewed, and Scopus-indexed publication venues, thereby supporting the credibility and reliability of the findings synthesized in this review. The distribution of publications by Country is shown in Fig. 3.

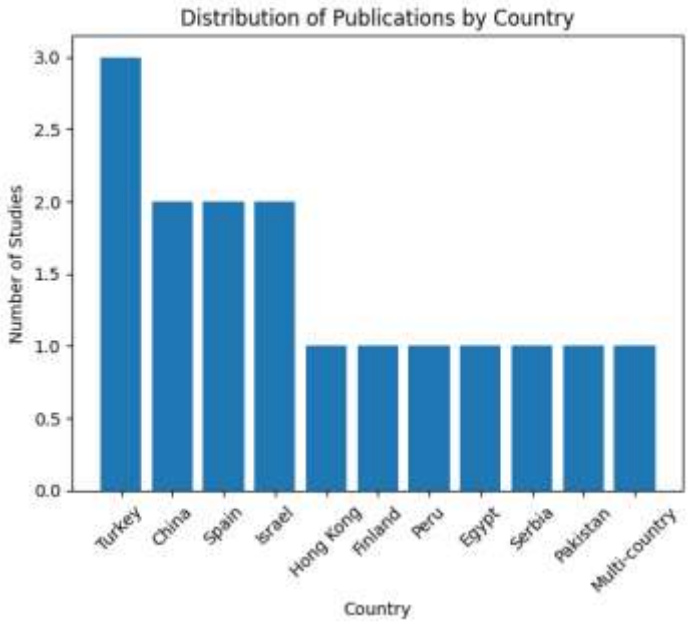


Fig. 3. Distribution of publications by Country

The distribution of publications by country shows that most studies were conducted in Turkey, followed by China, Spain, and Israel, each contributing a moderate number of studies. Other countries, including Hong Kong, Finland, Peru, Egypt, Serbia, and Pakistan, contributed a smaller number of publications. This indicates that research on Scratch-based learning is geographically diverse but still concentrated in specific regions.

Results and Discussions

Scratch has been widely adopted as a beginner-friendly, block-based visual programming language due to its ability to eliminate learning barriers such as complex text code syntax. To understand the effectiveness of this platform, Table 1 presents a comprehensive synthesis of 16 empirical and exploratory literature studies. This mapping highlights the diversity of educational contexts studied. This summary identifies the methodology, research focus, and key findings related to Scratch's pedagogical impact on students' cognitive development.

Table 1. Summary of empirical studies of Scratch-based programming

No	Focus	Key Findings	Ref. No.
1	Improving computational thinking (CT) and programming knowledge using a methodology based on metaphors (MECOPROG) and Scratch.	The metaphor-based Scratch approach significantly improved students' programming knowledge and CT scores.	[22]
2	Introducing CT with Scratch in mathematics teacher training to promote new learning strategies.	The experimental group using Scratch showed a significantly higher increase in CT scores compared to the control group.	[23]
3	The effect of game design with Scratch on students' problem-solving skills (self-confidence, self-control, avoidance).	Game design activities with Scratch significantly increased students' self-confidence in problem-solving compared to the regular IT curriculum.	[24]

No	Focus	Key Findings	Ref. No.
4	Understanding the relationship and causal influence between CT scores and computational participation (views, likes, remixes).	Identified 3 clusters of users. Interestingly, higher CT mastery scores of projects did not causally influence or lead to higher computational participation or popularity.	[25]
5	Exploring students' CT activities (planning, iteration, collaboration/remixing, debugging) during pair programming in Scratch.	Revealed imbalanced roles in pair programming and highlighted the need for pedagogical support to prevent trial-and-error debugging and ensure shared participation.	[26]
6	Integrating CT skills in a teacher education program starting from "scratch".	The course enhanced PSTs' appreciation for collaborative learning, facing challenges (debugging), and using Scratch for educational games, though they found CT challenging.	[27]
7	Assessing the usability and usefulness of "ScratchTSP", a custom Scratch extension designed to teach the Travelling Salesman Problem.	The ScratchTSP extension proved to be a user-friendly and effective tool that aided students in learning complex algorithms and improved analytical/problem-solving skills.	[28]
8	Analyzing the relationship/network between Scratch program components (aesthetic and operational criteria) and CT dimensions.	Aesthetic and operational criteria of Scratch are key driving components for developing computational practices, with the network structure showing no significant differences by gender.	[29]

A. RQ1: How does Scratch-based visual programming influence students' analytical thinking skills in ICT programming education?

Scratch-based visual programming directly improves students' analytical thinking skills and their subdimensions. Several studies provide evidence that Scratch-based visual programming contributes to the development of students' analytical thinking skills, either directly or indirectly. Experimental research has shown that Scratch education has a moderate to significant effect on developing students' analytical thinking skills [30]. Scratch influences analytical abilities through improvements in the following subdimensions: sorting, classifying information, comparing solutions, and making judgments or evaluations. Coding education lessons develop participants' analytical thinking skills. Improvements in sorting, classification, comparison, and evaluation contribute to improved classification, evaluation, and comparison skills, which are key indicators of analytical thinking.

There is a strong conceptual connection between Computational Thinking taught through Scratch and analytical thinking. Computational thinking is essentially defined as a form of analytical thinking[30]. The analytical component of CT is very similar to mathematical thinking for problem solving, engineering thinking through design concepts and process evaluation, and scientific thinking through systematic analysis[22]. Through Scratch, students learn to formulate problems, organize and analyze data logically, and evaluate and implement the most efficient solutions [31].

Scratch influences analytical thinking through the processes of problem decomposition and logical reasoning. When students engage in visual programming, they are confronted with problem-solving situations that require them to break down large, complex problems into smaller parts and create basic, generalizable solutions [32]. This process of placing these pictorial code blocks gradually stimulates the development of observation, analysis, logical thinking, and independent learning skills [24]. Both student and teacher perceptions and reflections explicitly confirm this analytical impact. When asked how Scratch impacted their thinking, 62% of students specifically stated that it improved their analytical thinking [32]. This finding is also strongly supported by teacher observations, who reported that during the visual programming learning process, students learned to see problems in greater detail, thereby developing their analytical, critical, and holistic thinking. These findings can be further explained through Bloom's Taxonomy, where analytical thinking is positioned as a higher-order cognitive skill involving analysis, evaluation, and synthesis [33]. Scratch-based learning supports these cognitive processes by enabling students to decompose problems, evaluate alternative solutions, and construct logical structures, thereby facilitating the transition from lower-order to higher-order thinking skills.

B. RQ2: What pedagogical and technological factors contribute to the development of students' analytical thinking skills when using Scratch in programming learning?

Pedagogical factors in Scratch-based learning include the use of metaphor-based approaches, collaborative learning, and structured instructional models. For example, the MECOPROG approach facilitates the understanding of abstract programming concepts by transforming them into concrete and visual representations, thereby enhancing conceptual clarity and reducing cognitive load [22]. This phenomenon can be seen in the use of the MECOPROG approach, which facilitates the understanding of abstract programming concepts by transforming them into concrete and meaningful representations using metaphors. MECOPROG connects complex programming constructs such as algorithms, loops, and conditional statements, which are translated into graphical blocks and interactive elements. This process allows students to transform abstract reasoning into executable visual forms, thereby reducing cognitive load and enhancing conceptual understanding. Learners can grasp the underlying concepts before engaging with formal coding structures.

Learning with Scratch allows students to work in groups or pairs [30]. This allows them to solve problems together, compare perspectives, discuss, and build on each other's ideas. Collaborative learning environments, such as pair programming, encourage students to exchange ideas, justify their reasoning, and construct shared solutions, which support deeper cognitive engagement [26]. Structured instructional models, such as the 5E model, further provide systematic opportunities for students to explore, explain, and refine their

understanding. These findings align with constructionist learning theory, which emphasizes active knowledge construction through meaningful artifact creation. Problem-based assignments or game design related to real-life situations significantly contribute to analytical thinking [24], [31]. Projects involving creating games using Scratch present students with unstructured problems that require them to continuously analyze, design algorithms, test, and revise their solutions [34]. The use of structured learning models, such as the 5E model (Engagement, Exploration, Explanation, Elaboration, Evaluation), fosters curiosity and provides systematic opportunities for students to explain their thinking processes [34]. Incremental support from teachers, such as breaking down complex coding patterns or providing guidance during debugging dead ends, is crucial to preventing students' analytical processes from devolving into inefficient guesswork or trial-and-error methods [26]. This finding aligns with the constructionist learning theory proposed by Papert, which emphasizes that learners construct knowledge more effectively when actively engaged in creating meaningful artifacts. Scratch-based learning environments enable students to design interactive projects, thereby promoting deeper cognitive engagement and analytical reasoning.

C. RQ3: How can Scratch-based learning be integrated into ICT curricula to enhance analytical thinking as a higher-order cognitive skill?

Programming education through Scratch directly and significantly improves students' analytical thinking skills as part of higher-order cognitive skills. Computational thinking taught through Scratch is essentially a form of analytical thinking[30]. Research shows that learning to program with Scratch has a moderate to strong effect on developing students' analytical thinking, as evidenced by significant improvements in the subdimensions of sequencing problem stages, classifying information, comparing solutions, and evaluating. Scratch requires students to formulate problems, manipulate data, and build algorithms, thus facilitating the development of other higher-order cognitive skills such as critical thinking, reflective thinking, and problem-solving [31]. Scratch can be effectively integrated into the compulsory Informatics course at the elementary, middle, or high school levels within the school curriculum. This integration is accomplished without imposing intricate text syntax rules on students; instead, it employs Scratch's visual block interface. To optimize the development of analytical thinking within the ICT curriculum, the learning framework can be integrated through the following strategies.

The implementation of project-based learning or problem-based learning models not only teaches computational theory but can also train students to create real products such as animations or video games [30], [32]. When students construct games with specific conditions and rules, they are forced to constantly analyze problems, break complex problems into smaller units (decomposition), and find and fix errors (debugging), which inherently train their

analytical skills. The use of the 5E instructional model (Engagement, Exploration, Explanation, Elaboration, Evaluation) that stimulates curiosity and exploration, or the Lifelong Kindergarten framework (Imagine, Create, Play, Share, Reflect) that balances the development of analytical cognitive skills with imagination [34]. Students are trained to move from the real world by understanding problems to the world of mathematics by creating mathematical models. The next step is to transform from mathematical models into the world of programming by drawing flowcharts, arranging code blocks, and viewing output [35]. Collaborative learning and pair programming train students to improve the construction of high-level knowledge based on Scratch through pair programming. Collaborating in programming requires students to compare points of view, negotiate, explain their logical thinking to peers, and plan code together, which is vital for developing analytical and problem-solving skills in a social context [26], [31]. Scratch-based learning can be effectively integrated into ICT curricula through structured and student-centered instructional strategies. At the classroom level, project-based learning can be implemented by engaging students in designing interactive products such as games or simulations that reflect real-world problems. Instructional frameworks such as the 5E model or the Lifelong Kindergarten approach can be used to guide students through stages of exploration, creation, and reflection. Collaborative approaches such as pair programming can be embedded into classroom activities to enhance communication and shared reasoning. At the curriculum level, Scratch can be introduced as an entry-level programming tool that emphasizes conceptual understanding through visual representation rather than syntactic complexity. This approach enables students to engage in meaningful problem-solving activities while gradually developing higher-order cognitive skills.

Implications of Findings

From a theoretical perspective, this study extends the understanding of the relationship between computational thinking and analytical thinking by positioning Scratch-based learning as a mediating mechanism for higher-order cognitive development. From a practical perspective, the findings suggest that educators should integrate Scratch into ICT curricula through structured pedagogical approaches such as project-based learning, collaborative programming, and scaffolded instruction. This integration enables students to engage in meaningful problem-solving activities that promote analytical reasoning and decision-making skills.

The findings of this study can be interpreted through several theoretical perspectives. First, based on Bloom's Taxonomy, analytical thinking represents a higher-order cognitive skill that involves analysis, evaluation, and synthesis. Scratch-based learning facilitates these

processes by enabling students to decompose problems, evaluate alternative solutions, and iteratively refine their designs.

From a pedagogical perspective, the effectiveness of Scratch aligns with constructionist learning theory, which emphasizes active knowledge construction through meaningful artifact creation. Additionally, Dual Coding Theory explains how the visual and interactive nature of Scratch enhances learning by combining graphical representations with logical reasoning. Finally, Cognitive Load Theory suggests that Scratch reduces extraneous cognitive load by eliminating syntactic complexity, allowing learners to focus on analytical problem-solving processes.

For curriculum designers, this review underscores the importance of rethinking ICT curriculum design by positioning Scratch-based learning as a vehicle for cultivating analytical thinking rather than merely teaching programming syntax. This requires a shift toward competency-based curriculum structures that integrate visual programming with inquiry-based and project-based pedagogies, supported by assessment frameworks that capture higher-order cognitive processes.

Conclusion

From a theoretical perspective, this study extends the understanding of the relationship between computational thinking and analytical thinking by positioning Scratch-based learning as a mediating mechanism for higher-order cognitive development. This claim is grounded in Papert's constructionist learning theory, which asserts that learners construct knowledge most effectively when actively engaged in creating meaningful, shareable artifacts. In the context of Scratch, the construction of interactive projects such as games, simulations, or animations requires students to externalize their thinking processes through code blocks, thereby transforming abstract analytical reasoning into executable, visible structures. This process of "thinking through making" operationalizes the transition from lower-order to higher-order cognition as described by Bloom's Taxonomy, specifically through the cognitive operations of analysis (decomposing problems), evaluation (comparing solution strategies), and synthesis (building algorithmic structures). Furthermore, Dual Coding Theory explains how Scratch's visual-block interface facilitates the simultaneous processing of graphical and logical representations, thereby reducing cognitive load and enabling learners to direct cognitive resources toward analytical reasoning rather than syntactic memorization.

From a practical perspective, the findings suggest that Scratch-based learning should be systematically integrated into ICT curricula through structured pedagogical strategies. First, educators can implement project-based learning by engaging students in designing interactive projects such as games or simulations, which require problem decomposition, logical reasoning, and iterative refinement. First, educators can implement project-based learning by

engaging students in designing interactive projects such as games or simulations, which require problem decomposition, logical reasoning, and iterative refinement. Empirical evidence supports that PBL in Scratch environments produces statistically significant improvements in students' analytical and logical thinking, and meta-analyses confirm that PBL significantly improves students' learning outcomes across programming contexts.

Second, collaborative learning approaches, such as pair programming, can be applied to encourage discussion, comparison of solutions, and shared problem-solving processes, thereby enhancing analytical thinking. _Research demonstrates that well-structured pair programming in higher education produces meaningful gains in collaborative problem-solving competence. Third, teachers should adopt scaffolded instruction by providing step-by-step guidance, debugging support, and reflective prompts to help students analyze their problem-solving strategies. Evidence shows that Computational Thinking Scaffolding (CTS) significantly improves students' higher-order problem-solving when integrated into programming activities without such scaffolding, students risk reverting to inefficient trial-and-error debugging, which limits the development of structured analytical processes.

At the curriculum level, Scratch can be embedded into ICT subjects as an introductory programming tool that emphasizes conceptual understanding rather than syntax complexity. This integration can be implemented through thematic or interdisciplinary learning, where students apply programming concepts to solve real-world problems in mathematics, science, or other domains. Therefore, Scratch-based learning provides a practical pathway for fostering analytical thinking as a core competency in ICT education.

Limitations and Future Research

This study has several limitations. First, the relatively limited number of articles included in the study was 16, which may affect the generalizability of the findings. Second, the databases used were only Scopus and ScienceDirect, potentially excluding relevant research from other databases. Third, most of the studies focused on computational thinking and problem-solving, with fewer studies explicitly addressing analytical thinking, which may impact the depth of the synthesis.

Future research should explore the direct impact of Scratch-based learning on analytical thinking through empirical and experimental studies across various educational levels. Further research is needed to develop a standardized instrument to measure analytical thinking in programming contexts. Future research could also explore the integration of Scratch with emerging technologies, such as artificial intelligence and adaptive learning systems, to enhance higher-order cognitive skills across various educational settings.

Acknowledgements

The author would like to express his gratitude for the support provided by the Institute for Research and Community Service, Sebelas Maret University, and for the financial support provided under contract number 460/UN27.22/PT.01.03/2026.

Declaration of Generative AI Use

The authors used Grammarly to assist in structuring the literature review and refining the grammatical flow. However, the conceptual framework, data analysis, and conclusions were independently developed and verified by the authors.

Conflict of Interest

The authors should declare that there is no conflict of interest.

References

[1] A. Dolmaci and F. E. Acar, "Development of a twenty-first-century skills curriculum through the flipped learning approach," *Humanit. Soc. Sci. Commun.*, pp. 1–18, 2025.

[2] M. Wang, M. E. E. Mohd Matore, and R. Rosli, "A systematic literature review on analytical thinking development in mathematics education: trends across time and countries," *Front. Psychol.*, vol. 16, no. June, pp. 1–10, 2025, doi: 10.3389/fpsyg.2025.1523836.

[3] R. C. Torres-Peña, D. Peña-González, J. L. Lara-Orozco, E. A. Ariza, and D. Vergara, "Enhancing Numerical Thinking Through Problem Solving: A Teaching Experience for Third-Grade Mathematics," *Educ. Sci.*, vol. 15, no. 6, pp. 1–27, 2025, doi: 10.3390/educsci15060667.

[4] H. Almazroa and W. Alotaibi, "Teaching 21st Century Skills: Understanding the Depth and Width of the Challenges to Shape Proactive Teacher Education Programmes," *Sustain.*, vol. 15, no. 9, 2023, doi: 10.3390/su15097365.

[5] R. Ardianti, A. Razak, H. Alberida, M. Fadilah, and M. Sari, "Analytical Thinking Skill and Problem Solving Skills Instruments in Biology Learning Using Rasch Model," *J. Educ. Res. Eval.*, vol. 8, no. 4, pp. 785–795, 2024, doi: 10.23887/jere.v8i4.84026.

[6] Suyatman, S. Saputro, W. Sunarno, and Sukarmin, "The Implementation of Research-Based Learning Model in the Basic Science Concepts Course in Improving Analytical Thinking Skills Suyatman," *Eur. J. Educ. Res.*, vol. 11, no. 1, pp. 403–421, 2021.

[7] Dinda Antika Putri, Sumarno, Berta Apriza, and Asri Widianingrum, "Analysis of the Effectiveness of Implementing Project-Based Learning Models on Students' Critical Thinking Skills in Elementary Schools: A Systematic Literature Review," *J. Lesson Learn. Stud.*, vol. 8, no. 1, pp. 129–142, 2025, doi: 10.23887/jlls.v8i1.90392.

[8] A. Nurjanah, D. Susilawati, J. M. Yusup, and A. Hendriyanto, "Use of Powtoon as a Technology-Based Creative Learning Medium: A Systematic Literature Review †," *Eng. Proc.*, vol. 107, no. 1, pp. 1–9, 2025, doi: 10.3390/engproc2025107054.

[9] T. S. Weng, "Animation and Manga on Improvement in Students' Problem-Solving Capabilities: Comparison of Two Psychometric Models," *Educ. Sci.*, vol. 14, no. 8, 2024, doi: 10.3390/educsci14080808.

[10] M. Wang, M. E. M. Matore, and R. Rosli, "Analytical Thinking in Mathematics: A Systematic Literature Review on Research Types and Content," *Int. J. Instr.*, vol. 18, no. 4, pp. 153–176, 2025, doi: 10.29333/iji.2025.1849a.

[11] A. S. Ramadani, Z. A. I. Supardi, Tukiran, and E. Hariyono, "Profile of Analytical Thinking Skills Through Inquiry-Based Learning in Science Subjects," *Stud. Learn. Teach.*, vol. 2, no. 3, pp. 45–60, 2021, doi: 10.46627/silet.v2i3.83.

[12] T. Regev Cohen, B. Armon, and A. Hershkovitz, "The Associations Between Computational Thinking and Learning to Play Musical Instruments," *Educ. Sci.*, vol. 15, no. 3, 2025, doi: 10.3390/educsci15030306.

[13] G. Aimicheva, G. Bekmanova, A. Omarbekova, A. Nazyrova, A. Khamzina, and Y. Kadyr, "A Digital Ecosystem Model for Developing Logical Thinking in Novice Programmers: Integrating Visualization Technologies and GenAI," *Technologies*, vol. 14, no. 1, p. 5, 2025, doi:

10.3390/technologies14010005.
[14] M. Lu and Z. Hu, "Digital Twin-Enhanced Programming Education: An Empirical Study on Learning Engagement and Skill Acquisition," *Computers*, vol. 14, no. 8, pp. 1–26, 2025, doi: 10.3390/computers14080322.
[15] A. H. Lubis, F. Yusup, M. D. Dasopang, and S. Januariyansah, "Effectivity of interactive multimedia with theocentric approach to the analytical thinking skills of elementary school students in science learning," *Prem. Educ. J. Pendidik. Dasar dan Pembelajaran*, vol. 11, no. 2, p. 215, 2021, doi: 10.25273/pe.v11i2.9658.
[16] J. Fagerlund, P. Häkkinen, M. Vesisenaho, and J. Viiri, "Computational thinking in programming with Scratch in primary schools: A systematic review," *Comput. Appl. Eng. Educ.*, vol. 29, no. 1, pp. 12–28, 2021, doi: 10.1002/cae.22255.
[17] H. Montiel and M. G. Gomez-Zermeño, "Educational challenges for computational thinking in k–12 education: A systematic literature review of 'scratch' as an innovative programming tool," *Computers*, vol. 10, no. 6, 2021, doi: 10.3390/computers10060069.
[18] N. I. Herawati, H. Kuswanto, M. Wahyuni, and A. Aristaria, "Scratch-Assisted Computational Thinking in Physics: A Literature Review," *JIPF (Jurnal Ilmu Pendidik. Fis.*, vol. 9, no. 1, p. 105, 2024, doi: 10.26737/jipf.v9i1.4696.
[19] M. L. Rethlefsen *et al.*, "PRISMA-S: An extension to the PRISMA statement for reporting literature searches in systematic reviews," *J. Med. Libr. Assoc.*, vol. 109, no. 2, pp. 174–200, 2021, doi: 10.5195/jmla.2021.962.
[20] M. J. Page *et al.*, "PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews," *BMJ*, vol. 372, 2021, doi: 10.1136/bmj.n160.
[21] M. J. Page *et al.*, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *PLoS Med.*, vol. 18, no. 3, p. e1003583, 2021, doi: 10.1371/journal.pmed.1003583.
[22] D. Pérez-Marín, R. Hijón-Neira, A. Bacelo, and C. Pizarro, "Can computational thinking be improved by using a methodology based on metaphors and Scratch to teach computer programming to children?," *Comput. Human Behav.*, vol. 105, no. December 2018, p. 105849, 2020, doi: 10.1016/j.chb.2018.12.027.
[23] Á. Molina-Ayuso, N. Adamuz-Povedano, R. Bracho-López, and M. Torralbo-Rodríguez, "Introduction to Computational Thinking with Scratch for Teacher Training for Spanish Primary School Teachers in Mathematics," *Educ. Sci.*, vol. 12, no. 12, 2022, doi: 10.3390/educsci12120899.
[24] O. Erol and N. S. Çırak, "The effect of a programming tool Scratch on the problem-solving skills of middle school students," *Educ. Inf. Technol.*, vol. 27, no. 3, pp. 4065–4086, 2022, doi: 10.1007/s10639-021-10776-w.
[25] B. Jiang, W. Zhao, X. Gu, and C. Yin, "Understanding the relationship between computational thinking and computational participation: a case study from Scratch online community," *Educ. Technol. Res. Dev.*, vol. 69, no. 5, pp. 2399–2421, 2021, doi: 10.1007/s11423-021-10021-8.
[26] J. Fagerlund, M. Vesisenaho, and P. Häkkinen, "Fourth grade students' computational thinking in pair programming with Scratch: A holistic case analysis," *Int. J. Child-Computer Interact.*, vol. 33, p. 100511, 2022, doi: 10.1016/j.ijcci.2022.100511.
[27] O. Broza, L. Biberman-Shalev, and N. Chamo, "'Start from scratch': Integrating computational thinking skills in teacher education program," *Think. Ski. Creat.*, vol. 48, no. November 2022, p. 101285, 2023, doi: 10.1016/j.tsc.2023.101285.
[28] U. U. Akbar, S. Akbar, M. Ali, R. Ullah, R. Khan, and I. O. Lopes, "Extending Scratch Framework to Improve the Analytical Skills of Undergraduate Students," *ACM Int. Conf. Proceeding Ser.*, no. March, pp. 15–19, 2024, doi: 10.1145/3674558.3674561.
[29] R. Oscco Solorzano, L. Torres Matos, D. Aroni Palomino, M. A. Alcca Huisa, J. H. Ppuño Huaricachi, and D. Merma Saico, "Analysis of the Scratch program for computational thinking in schoolchildren in an educational institution in Peru: A network analysis," *J. Educ. Technol. Dev. Exch.*, vol. 18, no. 3, pp. 118–132, 2025, doi: 10.18785/jetde.1803.06.
[30] B. Kocaman, "The Effect of Coding Education on Analytical Thinking of Gifted Students," *Int. J. Educ. Methodol.*, vol. 9, no. 1, pp. 95–106, 2023.
[31] B. Jiang and Z. Li, "Effect of Scratch on computational thinking skills of Chinese primary school students," *J. Comput. Educ.*, vol. 8, no. 4, pp. 505–525, 2021, doi: 10.1007/s40692-021-00190-z.
[32] S. Gökçe and A. A. Yenmez, "Ingenuity of scratch programming on reflective thinking towards problem solving and computational thinking," *Educ. Inf. Technol.*, vol. 28, no. 5, pp. 5493–5517, 2023, doi: 10.1007/s10639-022-11385-x.
[33] J. Sweller, J. J. G. Van Merrienboer, and F. G. W. C. Paas, "Cognitive Architecture and Instructional Design," *Educ. Psychol. Rev.*, vol. 10, no. 3, pp. 251–296, 1998, doi: 10.1023/A:1022193728205.

[34] X. Fang, D. T. K. Ng, W. T. Tam, and M. Yuen, "Integrating computational thinking into primary mathematics: A case study of fraction lessons with Scratch programming activities," *Asian J. Math. Educ.*, vol. 2, no. 2, pp. 220–239, 2023, doi: 10.1177/27527263231181963.
[35] S. Holstein and A. Cohen, "Scratch teachers' perceptions of teaching computational thinking with school subjects in a constructionist approach," *Think. Ski. Creat.*, vol. 56, no. January 2024, p. 101772, 2025, doi: 10.1016/j.tsc.2025.101772.

Authors



Risnawati is an undergraduate student in the Department of Informatics and Computer Engineering Education (PTIK) at Universitas Sebelas Maret (UNS), Indonesia. She is passionate about educational technology, digital learning, and computer-based instructional design. (email: risnawati_sso@student.uns.ac.id).



Aris Budianto is a lecturer at the Informatics and Computer Education Department, Faculty of Teacher Training and Education, Universitas Sebelas Maret, Surakarta, Indonesia. He is the Editor-in-Chief of The Journal of Informatics and Vocational Education (JOIVE). His research area is in Learning Technologies, Computer Vision, Artificial Intelligence in Education, Computer-Assisted Learning, and Educational Technology. (email: arisbudianto@staff.uns.ac.id).



Prof. Dr. Dedi Kuswandi is a lecturer at the State University of Malang, Indonesia, specializing in Educational Technology. He earned his undergraduate and master's degrees from IKIP Bandung and completed his doctoral degree in Curriculum and Educational Technology at the State University of Malang. Since 1990, he has been actively involved in teaching and research, focusing on innovative learning models and technology-enhanced education. (email: dedi.kuswandi.fip@um.ac.id).