

Course Title: Research Methodology in Computer Science

Course Code: AI800

Weighting: 20%

Lecturer: Dr. Mohammed Anbar

Hand-out Date: 20 June 2026

Hand-in Date: 4 July 2026

Hand-in Method: Online Submission via E-Learning.

Assignment Report: Literature Review on the Development of Computational Thinking Skills through Scratch and Block-Based Programming in Primary/Elementary Education

Table of contents:

- Summary research papers
- Critical Analysis
- Connections
- Conflicts between the findings
- Synthesis Matrix for Literature Review
- References

Summary research papers:

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

This pilot case study by Djambong and Freiman (2016) looked at how computational thinking (CT) skills grow in two groups of Canadian students who worked with two very different coding tools. Ten sixth graders used Scratch for one hour a week, and fourteen ninth graders used the LEGO EV3 robotics kit at one hour a day. Both interventions ran for about five weeks. To measure change, the authors gave 23 pre- and post-test tasks. Nineteen were taken from the Bebras contest and four were designed by the research team, spread over three difficulty levels. The gains were modest. Grade 6 moved from 29.6% to 33.0%, and Grade 9 from 44.9% to 48.2%. The older group did better overall, and progress was clearest on the easy tasks and on items that combined abstraction, decomposition and pattern recognition. Isolated algorithmic-thinking items barely moved and sometimes dropped slightly at post-test. The study has a few clear weaknesses. First, only 24 students took part and there was no random assignment, so the results cannot really be generalised. Second, the two groups used different tools at different ages, which mixes the effect of the environment with the effect of maturity. Third, the paper-and-pencil multiple-choice format makes it hard to see how students actually reason while they solve a task. (Djambong & Freiman, 2016)

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

Fagerlund, Häkkinen, Vesisenaho and Viiri (2021) put together a systematic review of the literature on assessing computational thinking through Scratch programming in K-9. They searched ScienceDirect, ERIC, SCOPUS, ACM and Google Scholar in late 2019, screened 432 records, and kept 30 after removing duplicates, off-topic papers and studies without a proper assessment tool. Their main contribution is a set of 12 core educational principles, or CEPs, that they see as the building blocks of CT in Scratch. These are abstraction, algorithms, automation, collaboration, coordination and parallelism, creativity, data, efficiency, logic, modelling and design, patterns and generalisation, and problem decomposition. The reviewed studies were sorted into four groups by the kind of programming substance they emphasised: code constructs (20 studies), coding patterns (10), programming activities (11), and other content (6). Assessment approaches ranged from artifact analysis and observation to discourse analysis, tests, self-evaluation and performance-based rubrics. The review has some limitations that the authors acknowledge. There is potential bias in how the research questions, search strategy and inclusion decisions were framed. The way CT is contextualised in the reviewed studies is not always consistent, and some CEPs (like Efficiency and Creativity) are rarely assessed with anything other than artifact analysis. Overall, the review shows that comprehensive and pedagogically meaningful assessment tools for CT at the primary level are still missing. (Fagerlund et al., 2021)

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

Piedade and Dorotea (2023) studied the effect of a school-year of Scratch activities on the CT skills of 189 fourth-graders from two Portuguese school clusters. The children were split into an experimental group of 96 and a control group of 93. Students in the experimental group had one hour of Scratch programming every week during the 2020-2021 school year, while the control group followed introductory computer-science activities. The instrument used to measure CT was the Beginners Computational Thinking Test (BCTt) of Zapata-Cáceres et al. (2020). It contains 25 multiple-choice items covering sequences, simple and nested loops, and if/else conditionals, and has a Cronbach's alpha of 0.85 in this sample. Results were clear. The experimental group scored $M=20.91$ ($SD=3.96$) on the post-test, compared with $M=16.25$ ($SD=4.06$) for the control group, a difference of 4.66 points that is

highly significant ($t(188)=7.99$, $p<0.001$). The gap was largest on simple loops (85% vs. 67%) and on if-then conditionals (79% vs. 58%). On the other hand, the study has some obvious weaknesses. There was no pre-test, so baseline differences between the groups cannot be ruled out. Sampling was non-probabilistic and limited to a single national context. Finally, the BCTt targets computational concepts but does not really cover practices or students' own perceptions of coding. (Piedade & Dorotea, 2023)

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

Martínez-Murciano and Pérez-Jorge (2024) reviewed the literature comparing Scratch and App Inventor as tools for developing CT-related competencies in primary education. Their PRISMA-guided search covered Web of Science (215 records), Scopus (190) and Dialnet (20), giving 425 initial results. After screening for language (English or Spanish), open access, primary-education focus and a publication window of 2018-2022, ten studies were retained for analysis. The main finding is that both tools develop very similar competencies. Computational thinking and motivation were addressed in 70% of the studies, collaborative work in 40%, and algorithmic thinking, problem-solving and creative thinking in about 20% each. The areas most often targeted were programming (90%) and mathematics (60%). Scratch was more often linked to gains in mathematical competence and reflective problem-solving, while App Inventor stood out for creativity and social skills. The review has clear limitations. Only 10 studies were kept, which is a very small final sample, and only one of them used App Inventor, so a true head-to-head comparison is not really possible. Several of the reviewed studies also had small or missing control groups (one had only 16 hours of App Inventor training), and the long-term durability of the observed effects is still an open question. (Martínez-Murciano & Pérez-Jorge, 2024)

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

Yusrina, Sutama, Sutopo and Muhibbin (2026) built and tested CT Math, a five-phase instructional model that uses Scratch to weave computational thinking into elementary mathematics lessons. The five phases are problem initialization, solution design, implementation, evaluation, and refinement and support. Design followed the ADDIE framework and was validated by a panel of 13 experts. The model was then tried out in three stages: a small pilot ($n=22$), a wider field trial ($n=51$), and a quasi-experimental comparison in which a CT Math class of 22 was compared with a control class of 21. Sessions ran on web-based Scratch, over four weeks and eight 70-minute meetings, and focused on plane-geometry problems (specifically, floor-tiling tasks). Normalized gain scores were 0.375 in the pilot, 0.587 in the field trial and 0.487 in the CT Math class, compared with 0.224 in the control class. The post-test difference between CT Math ($M=16.27$, $SD=2.31$) and control ($M=13.38$, $SD=3.44$) was significant ($t(41)=3.247$, $p=.002$), and the between-groups Hedges' g of 0.97 indicates a large effect. Classroom observations also showed clear improvements in decomposition, algorithm design and debugging. The main weaknesses are the narrow scope of the study (one Indonesian region, one math topic and only four weeks), some measurement problems (low inter-rater reliability in the language domain and a low baseline KR-20 of 0.411), and the lack of a formal fidelity-observation protocol. Non-random assignment and possible teacher-expectancy effects also limit causal claims. (Yusrina et al., 2026)

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

Koray and Bilgin (2023) asked whether Scratch coding activities, when placed inside the 5E learning cycle (Engage, Explore, Explain, Elaborate, Evaluate), can lift both computational thinking and block-based programming self-efficacy in a science classroom. Their study used a quasi-experimental single-group pretest-posttest design. Twenty-two sixth-grade students (12 girls and 10 boys) from a public middle school in the Eastern Black Sea region of Turkey took part during the 2022-2023 school year.

The intervention ran for five weeks (about 20 lessons) and covered the Force and Motion unit. Three instruments were used: the Computational Thinking Scale (Cronbach's alpha 0.809), a Self-Efficacy Perception Scale for Block-Based Programming (alpha 0.893), and an Activity Perception Scale, all analysed with paired-samples t-tests. Total CT scores rose significantly ($t(21)=-4.16$, $p<0.05$) with a mean gain of about 3.81 points. The improvement came mostly from Creativity and Collaboration; gains in Algorithmic thinking, Critical thinking and Problem-solving were not statistically significant. Self-efficacy scores went up from 30.73 to 33.45 ($t(21)=-7.09$, $p<0.05$), with large effect sizes ($d = 0.97$ to 1.88), and students clearly enjoyed the activities and felt they had grown personally. The main weaknesses are the small sample ($n=22$) with no control group, the fact that all students came from one school working on one science topic, and the very short (five-week) exposure. Because this was the students' first contact with Scratch, a novelty effect cannot be ruled out either. (Koray & Bilgin, 2023)

Critical Analysis:

Taken together, the six studies build a fairly convincing case that Scratch and other block-based environments can help develop CT skills at the primary and lower-secondary level. However, every one of them has its own methodological trade-offs. In terms of strengths, Piedade and Dorotea (2023) and Yusrina et al. (2026) both use quasi-experimental designs with control groups and validated CT instruments, and both produce sizeable effects (Hedges' g close to 1). The two systematic reviews (Fagerlund et al., 2021; Martínez-Murciano and Pérez-Jorge, 2024) look across a much wider body of work and offer conceptual maps of the field. Fagerlund and colleagues, in particular, propose the twelve Core Educational Principles that make CT easier to teach and to assess in a Scratch context. Koray and Bilgin (2023) and Yusrina et al. (2026) also go a step further than simple outcome measurement by embedding Scratch inside a formal pedagogical model (5E and ADDIE respectively), which gives the interventions a clearer instructional backbone.

At the same time, there are recurring weaknesses. Sample sizes are often small ($n=22$ in Djambong and Freiman, 2016, and again in Koray and Bilgin, 2023), and when comparison classes are used, students are almost never assigned to them randomly. Piedade and Dorotea (2023) collected only post-test data, which means baseline differences between the groups cannot be ruled out. Djambong and Freiman (2016) also compared two different tools with two different grade levels, so the effect of the tool cannot be separated cleanly from the effect of the students' age. Most of the empirical studies also run in one country and in one school, over relatively short windows of four to eight weeks, so it is hard to say much about longer-term retention. Both systematic reviews openly note that pedagogically meaningful and comprehensive assessment tools for CT at the primary level are still missing, and that softer principles such as Creativity and Efficiency are rarely assessed with anything beyond artifact analysis. The field would benefit from longer studies, replication in different cultural contexts, and richer process-based assessment that shows how students actually think while they code, rather than just what they answer at the end.

Connections:

- 1- Focus on Scratch and block-based programming: All six papers treat Scratch as the main instructional technology for teaching CT. Only Martínez-Murciano and Pérez-Jorge (2024) add App Inventor to the picture.
- 2- CT as the shared learning outcome: In every study, CT (usually operationalized around decomposition, abstraction, algorithms, patterns and debugging) is the outcome the researchers care about most.
- 3- Primary and lower-secondary focus: The studies cluster around Grades 4 to 9, roughly ages 9 to 14. This reflects a common belief that CT should be introduced early in schooling.
- 4- Similar research designs: Most of the empirical work uses pre/post-tests or quasi-experimental designs, and relies on validated CT instruments (BCTt, CTS, geometry tests) with t-tests or N-gain scores to quantify learning.
- 5- CT embedded in another subject: Several papers place Scratch inside another discipline, either mathematics (Yusrina et al., 2026) or science through the 5E cycle (Koray and Bilgin, 2023). The shared idea is that CT is best learned together with subject content, not on its own.
- 6- Shared concern about assessment: Both systematic reviews (Fagerlund et al., 2021; Martínez-Murciano and Pérez-Jorge, 2024) point out that comprehensive assessment tools for CT in primary schools are still missing, which fits with the empirical studies' reliance on relatively narrow multiple-choice tests.

Conflicts between the findings:

- 1- Which CT sub-skills actually improve: Piedade and Dorotea (2023) find significant gains on control-structure concepts (loops, conditionals). Yusrina et al. (2026) report gains in decomposition, algorithm design and debugging. Koray and Bilgin (2023), on the other hand, find significant improvements only in Creativity and Collaboration, and no significant change in Algorithmic thinking, Critical thinking or Problem-solving, which are exactly the skills the other studies highlight.
- 2- Visual programming versus tangible or robotics-based coding: Djambong and Freiman (2016) compare Scratch with LEGO EV3 and note that older students using the tangible tool do better. Other studies (Piedade and Dorotea, 2023; Yusrina et al., 2026) argue that Scratch on its own is enough to produce sizeable effects, and Martínez-Murciano and Pérez-Jorge (2024) suggest that Scratch and App Inventor develop almost the same competencies.
- 3- Single-tool focus versus cross-tool comparison: Most empirical papers stay with Scratch. Martínez-Murciano and Pérez-Jorge (2024) explicitly compare Scratch with App Inventor, and Djambong and Freiman (2016) with LEGO EV3. The empirical evidence for a clear "winner" between these environments remains inconclusive.
- 4- Curriculum integration versus stand-alone programming: Yusrina et al. (2026) and Koray and Bilgin (2023) embed Scratch inside a mathematics or science unit using a formal instructional model (ADDIE and 5E respectively), while Piedade and Dorotea (2023) treat Scratch as a stand-alone weekly programming lesson. All three approaches show positive results, but they disagree about how tightly Scratch should be tied to subject content.
- 5- Assessment strategy: The empirical papers lean heavily on multiple-choice CT tests such as the BCTt and CTS. Both systematic reviews argue that these instruments capture concepts but overlook creativity, efficiency and the process of problem-solving. This shows a real tension between what current instruments actually measure and what CT is supposed to include.

Synthesis Matrix for Literature Review:

Study	Authors	Year	Research Design	Key Findings / Results	Methodology	Limitations
Task-Based Assessment of Students' CT Skills Through Visual Programming or Tangible Coding Environments	Djambong & Freiman	2016	Exploratory pilot case study (pre/post-test)	Grade 9 significantly outperformed Grade 6 overall. Grade 6: 29.6% → 33.0%; Grade 9: 44.9% → 48.2%. Easy tasks improved most for Grade 6 (39.4% → 55.0%). Isolated algorithmic-thinking items showed limited or decreased post-test performance. Combined AB+DE+PR tasks scored highest.	24 students (Grade 6 n=10 Scratch; Grade 9 n=14 LEGO EV3). 5-week intervention, 2015–2016, New Brunswick, Canada. 23 CT tasks (19 Bebras + 4 team-designed). Descriptive quantitative analysis; no random assignment; no control group.	Very small sample (n=24) — not generalisable. Two different tools across two different grades (confounding). Paper-and-pencil multiple-choice format hides process. No control group; short 5-week window.
CT in Programming with Scratch in Primary Schools: A Systematic Review	Fagerlund, Häkkinen, Vesisenaho & Viiri	2021	Systematic literature review (K–9)	Defined 12 Core Educational Principles (CEPs) for CT in Scratch. Assessment methods vary: artifact analysis, observation, discourse analysis, tests, self-evaluation. Four programming substance categories: code constructs (20 studies), coding patterns (10), programming activities (11), other (6).	30 studies selected from 432 initial results. Databases: ScienceDirect, ERIC, SCOPUS, ACM, Google Scholar (Nov 2019). Empirical and non-empirical studies; iterative content evaluation.	Possible bias in RQ definition, search, and selection. Inconsistent contextualisation of CT concepts. Patterns, Generalisation and Decomposition not clearly operationalised. Efficiency and Creativity rarely assessed beyond artifacts. Comprehensive assessment tools for primary schools still lacking.
Effects of Scratch-Based Activities on 4th-Grade Students' CT Skills	Piedade & Dorotea	2023	Quasi-experimental, post-test only with control group	Experimental M=20.91 (SD=3.96) vs. control M=16.25 (SD=4.06); $\Delta M=4.66$. $t(188)=7.99, p<0.001$. Sequences 91% vs. 87%; simple loops 85% vs. 67%; nested loops 68% vs. 58%; conditionals if-then 79% vs. 58%. No significant gender or age effects.	189 4th-grade Portuguese students (exp n=96; control n=93), aged 9–10. Experimental group: weekly 1-hour Scratch across 2020–2021. Beginners CT Test (BCTt) — 25 multiple-choice items; $\alpha=0.85$. Post-test only; no random assignment.	No pre-test — baseline differences uncontrolled. Non-probability sampling. Assessed concepts only — not practices or perceptions. Single national context (Portugal).
CT with Scratch or App Inventor in Primary Education	Martínez-Murciano & Pérez-Jorge	2024	PRISMA systematic review	Both tools develop CT, motivation, algorithmic thinking, problem-solving, collaboration and creativity. Programming 90%, mathematics 60%, logic 30%, robotics 30%.	PRISMA review of 425 records (WoS 215, Scopus 190, Dialnet 20); 10 studies retained (2018–2022). Inclusion: English/Spanish, open access, primary education.	Only 10 studies retained. Some included studies had very small or absent control groups. Contradictory self-efficacy findings in one study.

Study	Authors	Year	Research Design	Key Findings / Results	Methodology	Limitations
				CT addressed in 70% of studies; motivation in 70%; collaborative work in 40%. Scratch → mathematical competence & reflective problem-solving; App Inventor → creativity & social skills.	Boolean search "Scratch OR App Inventor AND Education".	Long-term durability of effects unclear. Only one App Inventor study — head-to-head comparison limited.
CTMath: Enhancing Elementary Mathematics and CT Through a Structured Scratch-Based Model	Yusrina, Sutama, Sutopo & Muhibbin	2026	Design-based research + quasi-experimental	CTMath N-gain=0.487 vs. control 0.224. Post-test M=16.27 (SD=2.31) vs. 13.38 (SD=3.44); $t(41)=3.247$, $p=.002$. Between-groups Hedges' $g=0.97$ (large). Pilot N-gain=0.375; field trial=0.587. Gains in decomposition, algorithm design, debugging.	ADDIE-based five-phase model (Problem Init. → Solution Design → Implementation → Evaluation → Refinement). Expert validation (13 panelists); pilot $n=22$; field $n=51$; quasi-experimental $n=43$ (22 exp / 21 control). Fifth-grade Indonesian students; 4 weeks / 8 sessions of 70 min; web-based Scratch. Content: plane-geometry (floor tiling); 20-item test.	Single Indonesian region and single math topic (plane geometry). Short (4-week) intervention. Low inter-rater reliability (ICC) and low baseline KR-20 (0.411). No formal fidelity-observation protocol across teachers. Possible teacher-expectancy effects; non-random assignment.
The Effect of Block Coding (Scratch) Activities Integrated into the 5E Learning Model in Science Teaching on Students' CT Skills and Programming Self-Efficacy	Koray & Bilgin	2023	Quasi-experimental single-group pre-/post-test	Total CT Scale: $t(21)=-4.16$, $p<0.05$; mean gain +3.81. Creativity ($t=-2.83$) and Collaboration ($t=-3.04$) significant; $d=0.60-0.89$. Algorithmic thinking, Critical thinking and Problem-solving gains NOT significant. Self-Efficacy 30.73 → 33.45; $t(21)=-7.09$, $p<0.05$; $d=0.97-1.88$. High enjoyment and perceived personal-development ratings.	22 sixth-grade Turkish students (12 F, 10 M). 5-week intervention (20 periods) on "Force and Motion" science topic. 5E Model (Engage-Explore-Explain-Elaborate-Evaluate) integrated with Scratch. Instruments: CTS ($\alpha=0.809$), SPSRBP ($\alpha=0.893$), APS. Paired-samples t-tests; Shapiro-Wilk normality check.	Small sample ($n=22$). No control group. Single school; specific science topic only. Short 5-week duration. Students' first exposure to Scratch (novelty effect possible). Perception dip during difficult week-4 tasks.

References:

- 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).
- 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>
- 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>
- 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>
- 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>
- Yusrina, H., Utama, 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>